

MONTE-CARLO EVALUATION OF DIGITAL FILTERS
FOR FIRE CONTROL SYSTEMS

Michael Gordon Ketron

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA 93940

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

MONTE-CARLO EVALUATION OF DIGITAL FILTERS
FOR FIRE CONTROL SYSTEMS

by

Michael Gordon Ketron

June 1974

Thesis Advisor:

D. E. Kirk

Approved for public release; distribution unlimited.

T 16 1506

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) MONTE-CARLO EVALUATION OF DIGITAL FILTERS FOR FIRE CONTROL SYSTEMS		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis June 1974
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Michael Gordon Ketron		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE June 1974
		13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Naval Postgraduate School Monterey, California		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) Techniques are discussed for Monte-Carlo evaluation of several filters in a digital fire control system performing against an aircraft trajectory developed to simulate a diving attack. The programs developed are designed to allow for future modeling work in a three-dimensional theater of operations with expedient selection of the		

options that are discussed, and a comprehensive user's guide is provided. Comparison is made between a model of the filter currently used in the MK-86 Digital Fire Control System and some promising combinations of the several options available.

Monte-Carlo Evaluation of Digital Filters
for Fire Control Systems

by

Michael Gordon Ketron
Lieutenant (junior grade), United States Navy
B.S., University of Tennessee, 1971

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the
NAVAL POSTGRADUATE SCHOOL
June 1974

ABSTRACT

Techniques are discussed for Monte-Carlo evaluation of several filters in a digital fire control system performing against an aircraft trajectory developed to simulate a diving attack. The programs developed are designed to allow for future modeling work in a three-dimensional theater of operations with expedient selection of the options that are discussed, and a comprehensive user's guide is provided. Comparison is made between a model of the filter currently used in the MK-86 Digital Fire Control System and some promising combinations of the several options available.

TABLE OF CONTENTS

I.	INTRODUCTION -----	10
II.	PROBLEM DESCRIPTION -----	12
	A. KALMAN FILTER THEORY -----	12
	B. SYSTEM MODELS -----	16
	C. MONTE-CARLO SIMULATION -----	19
III.	DEVELOPMENT OF MONTE-CARLO SIMULATION PROGRAM (MCSP) -----	27
	A. TRACK MODEL -----	27
	B. TRACK OPTIONS -----	39
	C. SHELL-AT-TARGET ACCURACY -----	47
	D. ONLINE CALCULATION OF COVARIANCE MATRIX OF POSITION MEASUREMENT NOISE -----	50
	E. COVARIANCE MATRIX OF STATE EXCITATION -----	55
	F. INITIALIZATION -----	60
IV.	DESCRIPTION OF MONTE-CARLO SIMULATION PROGRAM (MCSP) -----	63
	A. APPLICATION -----	63
	B. INPUTS -----	63
	C. OUTPUT -----	69
V.	SUMMARY OF REPRESENTATIVE SIMULATION RUNS -----	71
	A. DESCRIPTION OF RESULTS -----	71
	B. EVALUATION OF RESULTS -----	74
VI.	CONCLUSIONS AND RECOMMENDATIONS -----	80
APPENDIX A:	PROGRAM LISTING FOR MONTE-CARLO RUN REQUIREMENT EVALUATION -----	82
APPENDIX B:	TRACK GENERATING PROGRAM -----	85

APPENDIX C:	LISTING OF MONTE-CARLO SIMULATION PROGRAM (MCSP) -----	93
APPENDIX D:	LISTING OF SHELL FLIGHT TIME INTERPOLATION PROGRAM -----	122
APPENDIX E:	SAMPLE SIMULATION -----	126
BIBLIOGRAPHY	-----	147
INITIAL DISTRIBUTION LIST	-----	148

LIST OF TABLES

I.	Comparison of Empirical and Theoretical Normally Distributed Random Numbers -----	25
II.	Estimator Performance Table -----	75
III.	Number of Hits Within Several Miss Distances for Selected Filters -----	76

LIST OF FIGURES

Figure

1.	Block Diagram of the Discrete Plant and Kalman Filter -----	15
2.	Coordinate System -----	17
3.	$t - \beta$ Diagram -----	22
4.	Number of Samples Required to Estimate the Standard Deviation Within P Percent of its True Value With Confidence Coefficient γ -----	23
5.	Geometry for Turn with Velocity Vector Initially Parallel to Axis -----	31
6.	Geometry for Right and Left Turns with Velocity Vector Initially to Right and Left of Axis, Respectively -----	33
7.	Geometry for Right and Left Turns with Velocity Vector Initially to Left and Right of Axis, Respectively -----	34
8.	Initial Angle, τ , to Right of Axis - Right Turn -----	36
9.	Initial Angle, τ , to Right of Axis - Left Turn -----	36
10.	Initial Angle, τ , to Left of Axis - Right Turn -----	37
11.	Initial Angle, τ , to Left of Axis - Left Turn -----	37
12.	Constant Acceleration Motion in a Straight Line -----	40
13.	Constant Radial Acceleration Turn with Climb Angle of β -----	40
14.	Aircraft Motion in XZ Plane -----	41
15.	Aircraft Motion in XY Plane -----	42
16.	Rotation of Track -----	44

17.	Velocity Vectors for Track Rotation -----	45
18.	Shell-at-target Accuracy Diagram -----	48

I. INTRODUCTION

With the increasing cost and complexity of modern gunfire control systems there exists a correspondingly increasing need for thorough evaluation of simulated performance with accurate models before major commitment is made.

Specifically, the MK-86 Digital Gunfire Control System has come under such scrutiny and this thesis is one in a series designed to define, evaluate and offer alternative approaches to the existing system. While the original motivation for the study in this thesis was to explore alternative techniques to those used in the existing system, the resulting derivations and computer programs lend themselves to other system modeling situations within certain restrictions. The development of this thesis is intended to allow future modeling work to easily take advantage of the results contained herein.

The pattern followed in a typical cycle of the gunfire control system is the acquisition of data by a radar, analysis of the measurements in a filtering section, and the generation of gun orders in a third general section. The entire model should study all of these segments, but in order to simplify the problem each segment may be studied separately. This thesis is designed for studying the filtering section and assumes that input data is available from a

typical radar and that the results would be used by another section to generate the gun orders.

In order to evaluate possible filtering methods for analyzing the radar data it was necessary to generate realistic, simulated radar measurements for the model. Therefore, a hypothetical track was developed with the capacity for variation to provide the necessary input data. In addition, the various special features that are presented in this thesis were organized in the computer simulation for straightforward selection and implementation thus offering maximum flexibility for evaluation of the various filtering methods. This designed-in choice of options was intended, also, to lend the general program to other applicable situations.

II. PROBLEM DESCRIPTION

A. KALMAN FILTER THEORY

Sequential estimation is characterized by the serial recursive processing of observations taken in time sequence. The result of every processing cycle is the current best estimate of the vector being estimated. This estimate therefore embodies all observation data up to and including the current observation. As a new observation is made, the current estimate is updated to reflect this most recent data. In such an estimation scheme the calculations are identical in nature from cycle to cycle so they are ideally suited for implementation on a digital computer.

The Kalman filter [1] is a recursive filter of the type applicable to a digital fire control system in which discrete observations are available from the radar. The filter offers the capability of not only generating estimates of the observed system's states but, also, of predicting future system (or plant) states.

The linear discrete system for which the Kalman filter is designed is characterized by the state and output equations

$$\underline{x}(k+1) = \underline{\phi} \underline{x}(k) + \underline{\Delta} \underline{u}(k) + \underline{\Gamma} \underline{w}(k) \quad (1)$$

$$\underline{z}(k) = \underline{H} \underline{x}(k) + \underline{v}(k) \quad (2)$$

where

$\underline{x}(k)$ is the n-dimensional state vector at time $t = kT$,

$\underline{u}(k)$ is the p-dimensional deterministic input vector at time $t = kT$,

$\underline{z}(k)$ is the m-dimensional vector of measurements or observations taken at time $t = kT$,

$\underline{w}(k)$ and $\underline{v}(k)$ are q-dimensional and m-dimensional noise processes, respectively, at time $t = kT$,

ϕ is the $n \times n$ state transition matrix--assumed to be known,

H is the $m \times n$ observation matrix--assumed to be known,

Δ and Γ are $n \times p$ and $n \times q$ matrices, respectively, which relate the deterministic and non-deterministic forcing terms to the state vector --they are assumed to be known,

T is the time between measurements, and k is a non-negative integer.

The noise statistics are summarized below

$$E[\underline{v}(k) \underline{v}^T(j)] = \underline{R}(k) \underline{\delta}(k, j) \quad (3)$$

$$\underline{\Gamma} E[\underline{w}(k) \underline{w}^T(j)] \underline{\Gamma}^T = \underline{Q}(k) \underline{\delta}(k, j) \quad (4)$$

$$E[\underline{v}(k) \underline{w}^T(j)] = 0 \text{ for all } (k, j) \quad (5)$$

$$\underline{\delta}(k, j) = \begin{cases} 0 & k \neq j \\ 1 & k = j \end{cases} \quad (6)$$

It is assumed that the initial state is a random variable with known mean and covariance

$$E[\underline{x}(0)] = \underline{\hat{x}}_0, \quad E\{[\underline{x}(0) - \underline{\hat{x}}_0][\underline{x}(0) - \underline{\hat{x}}_0]^T\} = \underline{P}_0 \quad (7)$$

In addition, it is assumed that the measurement noise and initial state are uncorrelated

$$E[\underline{x}(0) \underline{v}^T(k)] = 0 \text{ for all } k \quad (8)$$

and that the forcing noise and initial state are uncorrelated

$$E[\underline{x}(0) \underline{w}^T(k)] = 0 \text{ for all } k \quad (9)$$

The Kalman filter equations are summarized below

$$\underline{G}(k) = \underline{P}(k/k-1) \underline{H}^T [\underline{H} \underline{P}(k/k-1) \underline{H}^T + \underline{R}(k)]^{-1} \quad (10)$$

$$\underline{P}(k/k) = (\underline{I} - \underline{G}(k) \underline{H}) \underline{P}(k/k-1) \quad (11)$$

$$\underline{P}(k+1/k) = \underline{\phi} \underline{P}(k/k) \underline{\phi}^T + \underline{Q}(k) \quad (12)$$

$$\underline{\hat{x}}(k/k) = \underline{\hat{x}}(k/k-1) + \underline{G}(k) [\underline{z}(k) - \underline{H} \underline{\hat{x}}(k/k-1)] \quad (13)$$

$$\underline{\hat{x}}(k+1/k) = \underline{\phi} \underline{\hat{x}}(k/k) + \underline{\Delta} \underline{u}(k) \quad (14)$$

where the notation $(k/k-1)$ is defined as a condition at time $t = kT$ given information up to and including time $t = (k-1)T$.

The matrices in these equations are

$\underline{G}(k)$ - $n \times m$ gain matrix

$\underline{R}(k)$ - $m \times m$ covariance matrix of measurement error

$\underline{P}(k/k)$ - $n \times n$ covariance matrix of estimation error

\underline{I} - $n \times n$ identity matrix

$\underline{P}(k+1/k)$ - $n \times n$ prediction covariance matrix

$\underline{Q}(k)$ - $n \times n$ covariance matrix of state excitation defined as $\underline{\Gamma} E[\underline{w}(k) \underline{w}^T(k)] \underline{\Gamma}^T$

$\underline{\hat{x}}(k/k)$ - $n \times 1$ optimal (minimum variance) estimates of $\underline{x}(k)$

$\underline{z}(k)$ - $m \times 1$ observation vector

$\underline{\hat{x}}(k+1/k)$ - $n \times 1$ optimal predicted value of $\underline{x}(k)$.

A block diagram of the discrete plant and Kalman filter is shown in Figure 1.

The Kalman filter takes advantage of all previous state measurements, \underline{z} , along with their respective error estimates,

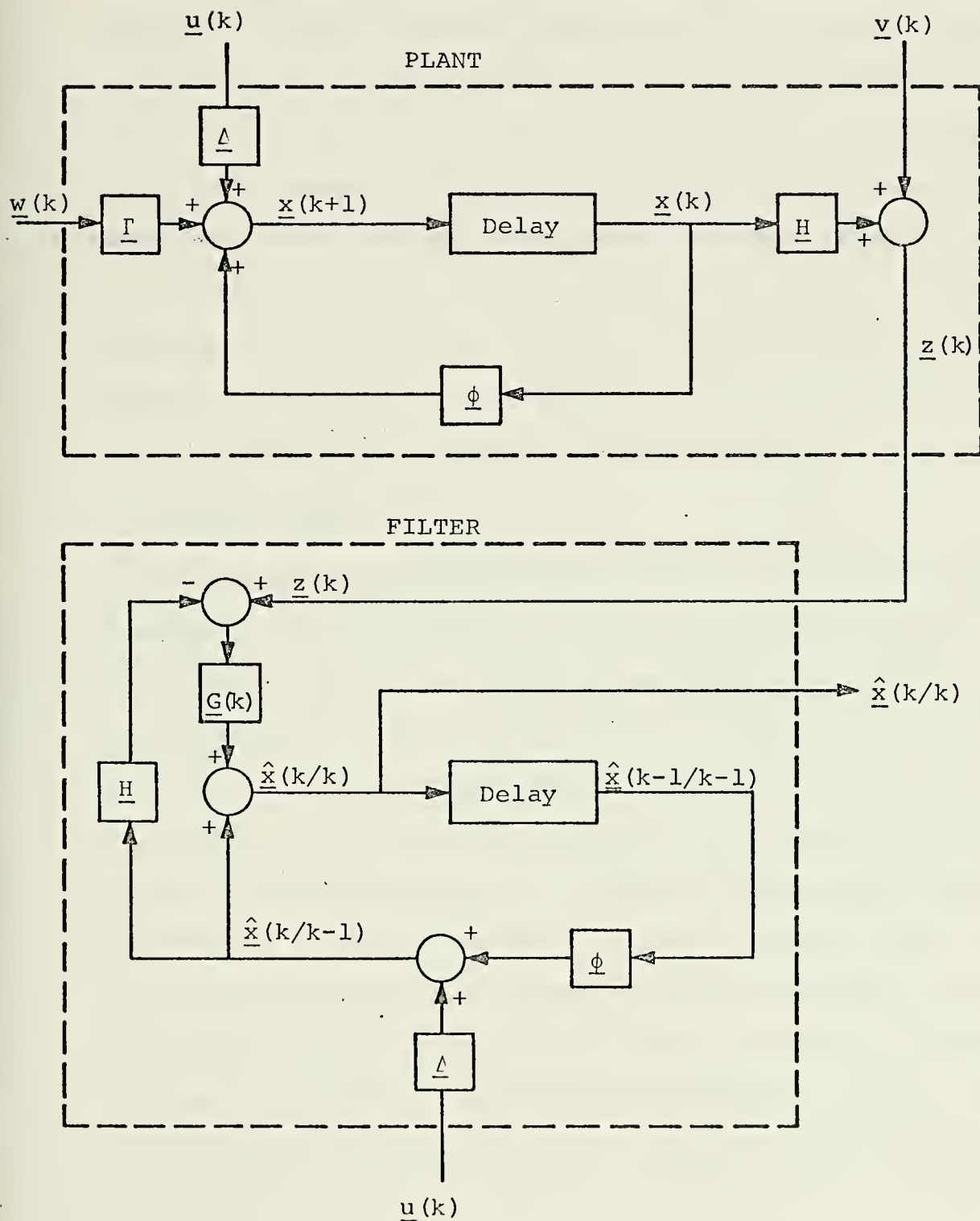


Figure 1. Block Diagram of the Discrete Plant and Kalman Filter.

and predicts ahead what the system states should be based on the state transition matrix, ϕ , and any known deterministic forcing \underline{u} . When a new measurement becomes available for current state estimation the filter takes the predicted state vector from the previous iteration, $\hat{\underline{x}}(k/k-1)$, and corrects it by some amount depending on the difference between the predicted vector and the measurement vector. Normally the amount of correction is a fraction (less than one) of the difference and is defined by the gain matrix, $\underline{G}(k)$, which has been calculated using equations (10) - (12) so that the state estimates yield minimum variance estimates.

B. SYSTEM MODELS

In order to apply the Kalman filter to a given situation a model for the plant must be specified. In the case of a fire control system the plant is defined as the target. The motion of the target is characterized by quantities such as position, velocity, acceleration, acceleration rate, etc.

For most applications the plant can be approximated as either a constant-velocity or constant-acceleration target. In terms of a three-dimensional tracking problem with filtering in an orthogonal coordinate system the resultant state vector for the constant-velocity model includes six states --three for position definition and three for velocity. The constant-acceleration model also includes three acceleration states.

In this thesis, for example, the filtering is done in an XYZ Cartesian coordinate system, shown in Figure 2, using

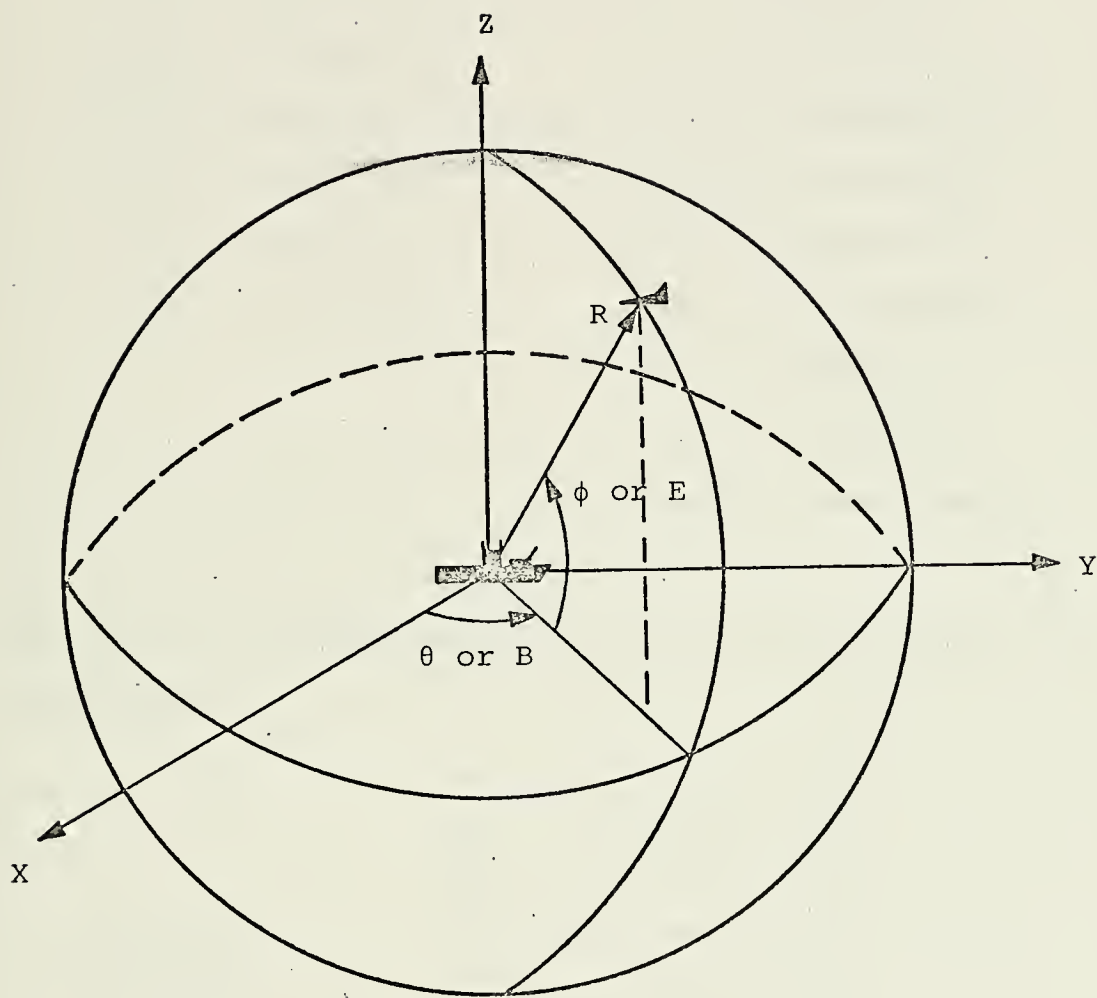


Figure 2. Coordinate System.

both constant-velocity and constant-acceleration models with the resultant state vectors

Constant-velocity Model

x_1 x position
 x_2 x velocity
 x_3 y position
 x_4 y velocity
 x_5 z position
 x_6 z velocity

Constant-Acceleration Target

x_1 x position
 x_2 x velocity
 x_3 x acceleration
 x_4 y position
 x_5 y velocity
 x_6 y acceleration
 x_7 z position
 x_8 z velocity
 x_9 z acceleration

The state transition matrices, $\underline{\phi}$, for the two models are shown below

$$\begin{array}{l} \text{Constant-} \\ \text{Velocity} \end{array} \quad \underline{\phi} = \begin{bmatrix} 1 & T & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & T & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & T \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (17)$$

$$\text{Constant Acceleration } \underline{\phi} = \begin{bmatrix} 1 & T & \frac{T^2}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & T & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & T & \frac{T^2}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & T & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & T & \frac{T^2}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & T \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (18)$$

For the models used in defining the plant in this thesis the deterministic and non-deterministic forcing terms, $\underline{\Delta} \underline{u}(k)$ and $\underline{\Gamma} \underline{w}(k)$, are assumed to be zero.

C. MONTE-CARLO SIMULATION

The basic premise upon which computer simulation is justified is that the model used for study accurately portrays the phenomenon as it occurs in practice. In order to assure that this requirement is satisfied repeated independent experiments yielding classical random-sample statistics are necessary. The requirement as it applies to actual simulation exercises implies that those parameters in the model that are variable require alteration in a random manner dictated by the expected randomness in practice.

Monte-Carlo evaluation is the process of satisfying the need for randomness in some of the model parameters and is especially applicable with computer simulation where the

necessary calculating capacity and speed is available. Further discussion of Monte-Carlo techniques and applications may be found in [2].

The following discussion presents the necessary information for deciding how many Monte Carlo runs, or iterations, are required to satisfy some desired statistical confidence in the simulation. The source of theoretical information against which empirical sampling of available random number generators are compared is [3].

The use of available random number generators, such as GAUSS or NORMAL for normally distributed random numbers, is the most straightforward method for obtaining the desired randomness to introduce into the simulation. Two factors that must be considered in evaluating the performance of a random number generator are (1) the statistics of the empirical samples compared with the desired statistics and (2) the confidence for which the results of (1) are guaranteed.

For the type of random numbers needed with the Monte-Carlo simulation of the filtering process in a fire control system the merit of acceptance is based on the actual mean and standard deviation (or variance) of the normally distributed random numbers compared to the desired, or specified, values of these quantities. Considering the mean of the distribution first, several variables can be defined that are used in calculating a desired number of samples to achieve an acceptable mean. These include

- d - a margin of acceptable error of the sample mean expressed as a percentage of the desired standard deviation, such as 0.1 (10% of the standard deviation),
- α - the risk involved in the result, also signifying (100 - α) per cent confidence in the result,
- β - defined as $(1 - \alpha/2)$.

Entering Figure 3 with the value of β the variable t is defined. The number of samples necessary to give a sample mean within d of the desired mean with a known risk of α is given by

$$n = \frac{t^2 \sigma^2}{d^2} \quad (19)$$

It can be seen that if d is expressed as a percentage of σ , the actual value of the standard deviation is not required to solve for the number of samples needed.

To determine the number of samples necessary for desired statistics of the standard deviation refer to Figure 4. Here again, the margin of acceptable error is expressed as a percentage of the standard deviation, and this percentage is the parameter P scaled along the abscissa. The ordinate is labeled "degrees of freedom" and represents the number of samples needed to achieve the various confidences plotted as bars across the graph.

Working through an example shows how the curves are used. Suppose it is desired to execute enough Monte-Carlo runs to assure 90% confidence in sample statistics having an actual mean and an actual standard deviation within 10% of its desired value. Considering the mean first:

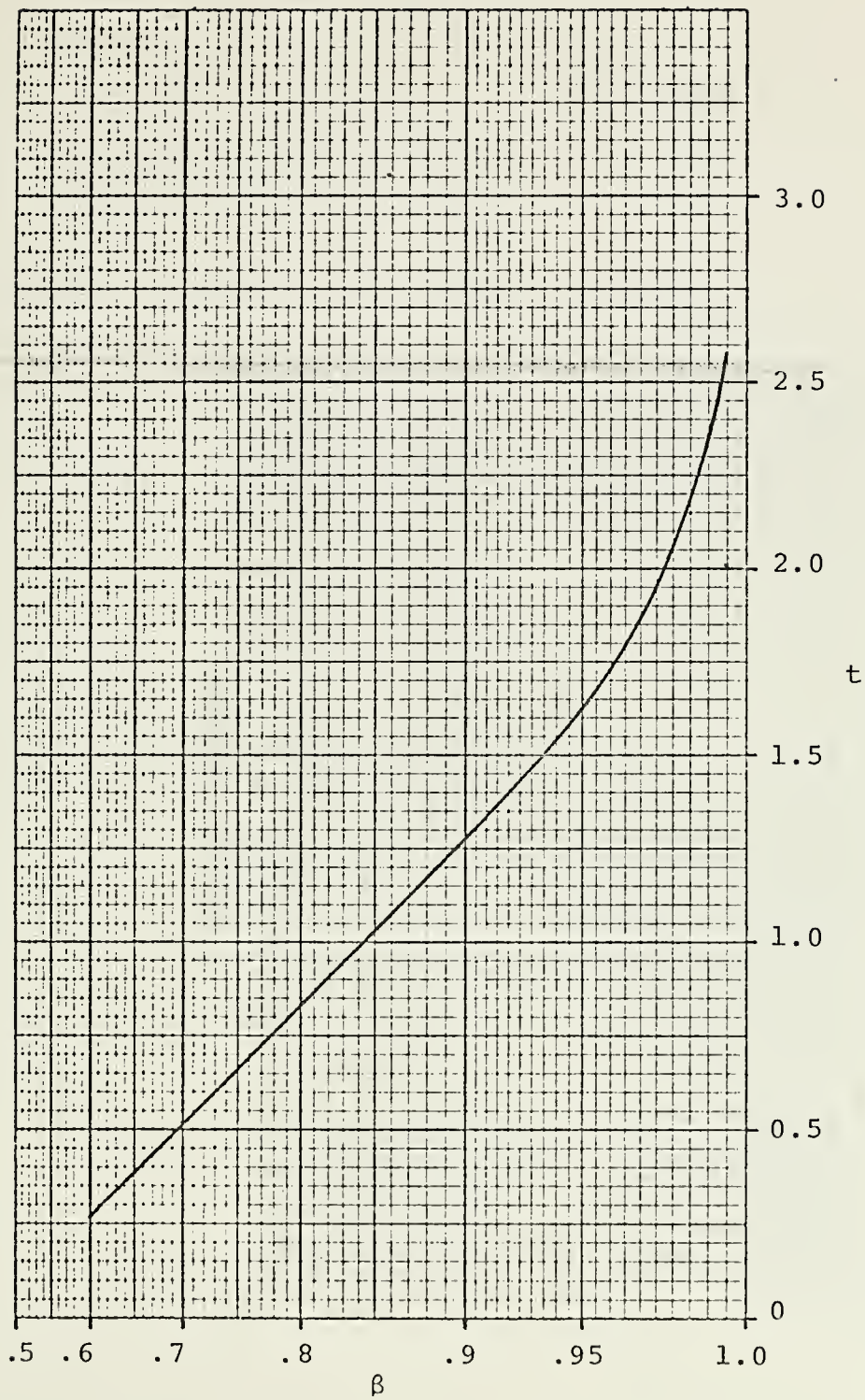


Figure 3. $t - \beta$ Diagram.

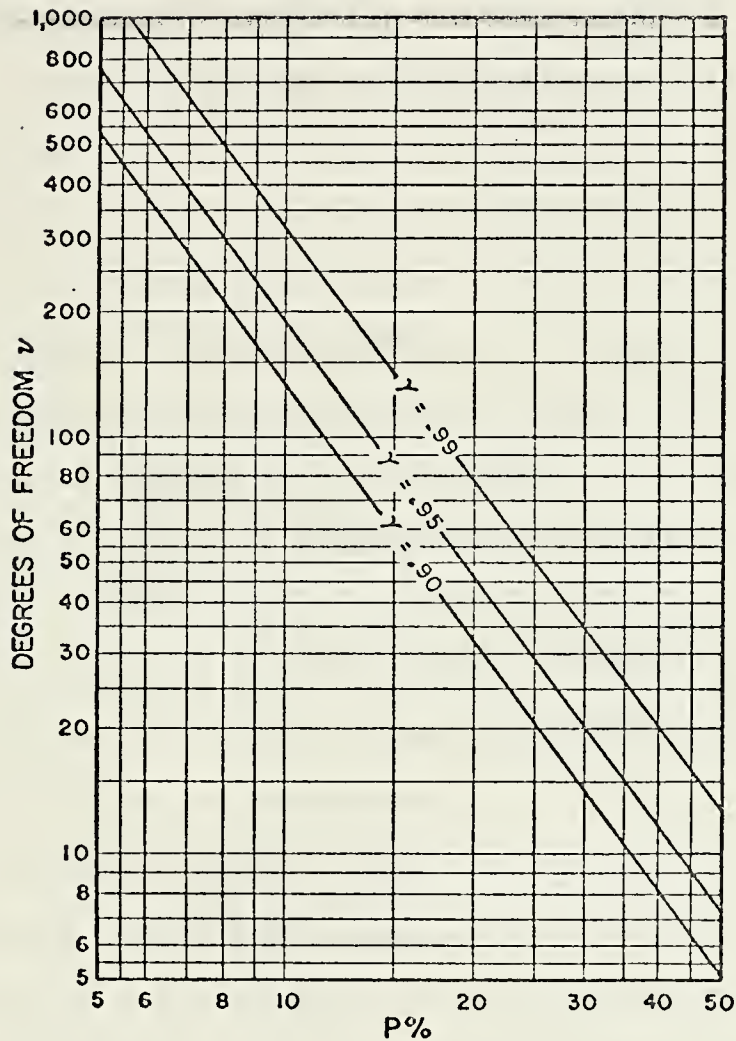


Figure 4. Number of Samples Required to Estimate the Standard Deviation within P Percent of Its True Value with Confidence Coefficient γ .

$$\beta = 1 - \frac{\alpha}{2} = 1 - \frac{.1}{2} = .95 . \quad (20)$$

From Figure 3 the value of t is found to be 1.645, and

$$n = \frac{(1.645)^2 \sigma^2}{(.1)^2 \sigma^2} = 270 . \quad (21)$$

Referring to Figure 4 to find the number of samples required for the desired standard deviation results gives $n = 140$. Therefore, in order to satisfy all restrictions 270 samples would be required.

Verification of these theoretical results involved sampling two subroutines available in the NPS Computer Facility Sublibrary, GAUSS and NORMAL. The approach used was to sample each number generator for 1000 ensembles of several numbers of samples in each ensemble. The number 1000 was chosen to obtain steady-state values in the statistics determined for the runs of length 50, 75, 100, 150, 200, 500, 700, 1000, and 5000 samples each. A summary of the results is included as Table I,¹ and the FORTRAN program for the study may be found in Appendix A.

It may be noted in Table I that the results for the normally distributed random number generator GAUSS are incomplete, and this is a consequence of the substantially greater execution time required than for NORMAL. Whereas NORMAL used

¹Theoretical results are limited in accuracy to one's ability to read Figure 3. Also, percent of standard deviations within criteria of desired standard deviations are not shown for theoretical calculations because (1) Figure 4 is not complete and (2) the limiting factor in satisfactory M-C simulation is the satisfaction of requirements for the mean, not for the standard deviation.

TABLE I
COMPARISON OF EMPIRICAL AND THEORETICAL
NORMALLY DISTRIBUTED RANDOM NUMBERS

Number of Samples	Percent Criteria	Theoretical \bar{M}^a	NORMAL		GAUSS	
			\bar{M}^a	SD^b	\bar{M}^a	SD^b
50	5	28.0	27.0	32.7	27.4	37.2
	10	52.4	51.9	63.7	52.4	64.8
	15	71.2	69.0	82.8	72.3	83.9
	20	84.2	83.9	94.3	85.7	94.5
	25	92.4	92.9	98.1	94.2	98.4
75	5	33.6	33.7	42.2	34.7	44.2
	10	62.0	60.4	75.1	60.6	76.6
	15	80.8	80.2	82.5	81.3	93.2
	20	91.6	91.9	98.5	92.0	99.0
	25	96.8	97.0	99.8	97.6	99.9
100	5	38.6	39.0	49.8	37.3	50.7
	10	68.0	67.9	83.2	66.9	83.3
	15	86.8	86.3	96.0	87.1	97.2
	20	95.4	95.1	99.2	95.7	99.6
	25	98.8	98.3	100.0	99.1	100.0
150	5	47.0	45.2	60.0	45.3	60.8
	10	78.0	78.3	89.6	78.2	92.1
	15	93.4	91.9	99.0	94.2	99.3
	20	98.4	98.0	99.9	99.0	100.0
	25	99.8	99.7	100.0	99.9	100.0
200	5	52.0	52.7	66.8	52.6	68.7
	10	84.2	82.2	95.3	83.6	95.2
	15	96.4	96.0	99.8	96.3	99.9
	20	99.6	99.4	100.0	99.1	100.0
	25	100.0	99.9	100.0	99.9	100.0
500	5	73.4	75.3	87.0	72.6	88.9
	10	97.0	96.7	99.9	97.0	99.9
	15	100.0	99.9	100.0	99.9	100.0
	20	100.0	100.0	100.0	100.0	100.0
700	5	81.4	81.0	93.5	81.5	93.7
	10	99.2	99.6	100.0	98.6	100.0
	15	100.0	100.0	100.0	100.0	100.0
1000	5	89.0	90.4	96.2	-	-
	10	100.0	99.9	100.0	-	-
	15	100.0	100.0	100.0	-	-
5000	5	100.0	100.0	100.0	-	-

Note (a) - Percent of Means within percent criteria of desired standard deviation

Note (b) - Percent of Standard Deviations within same percent criteria as (a)

approximately 25 minutes of execution time to carry out the entire analysis, GAUSS required 90 minutes to evaluate up through 700 sample ensemble runs. For this reason NORMAL is recommended when normally distributed random numbers are desired.

III. DEVELOPMENT OF MONTE-CARLO SIMULATION PROGRAM (MCSP)

A. TRACK MODEL

In order to provide realistic data for testing filter performance against an airborne target it was necessary to develop a model based on a conventional dive bomb approach by a modern aircraft against a seaborne target. From knowledgeable sources the parameters of such an approach were obtained and, in accordance with the 4-Hz sampling rate of the radar, used to develop a simulation including such features as a constant-acceleration dive, a pull-out following ordinance release, and evasive maneuvers while withdrawing in a gradual climb. The individual segments of the 233-point track will be discussed as well as the methods one may use to develop other tracks.

While in actual operation a radar measures range, bearing, and elevation (RBE), and perhaps rates as well, convenience dictated development of the track in an XYZ coordinate system centered at the ship, or radar,² as shown in Figure 2. Such parameters as ship's speed, bearing, roll, pitch, etc. were not considered in the simulation to avoid model complexity. These quantities, while offering potential sources of error to any overall tracking system, were avoided to concentrate on the filter performance and not the various correction factors applied to input measurement data.

²The radar and the gun are assumed to be at the center of the ship, or origin of coordinate system, and at sea level.

Satisfying the requirement for randomness in the track data points, the variable factors mentioned earlier for Monte-Carlo simulation, involved, when using one basic track, the addition of randomly distributed Gaussian measurement noise to every point. An alternative approach might have been to run the simulation with many independent tracks randomly distributed about the expected track, but this would have caused much additional complexity and increased execution time in the simulation.

In order to realistically generate noisy measurements the track's XYZ data points, after being read by the Monte-Carlo Simulation Program (MCSP), are converted to RBE, the desired Gaussian measurement noise with zero mean and appropriate standard deviation is added to each data point, conversion back into the same XYZ coordinate system takes place, and the data is available as simulated coordinate-converted output from the radar ready for analysis.

The original track is layed out with the aircraft approaching from the +X direction with initial radar contact initiated at a range of 7200 yards and altitude of 4333 yards. Its initial velocity is approximately 250 knots, and in the scenario of the attack has just climbed to the given altitude from a low or medium level approach in preparation for the dive phase of the attack. A summary of the track segments follows:

Track Points	Segment Description	G Turn	Comments
1 - 33	Initial Dive	1	Assumes 40° Dive Angle
34 - 113	Accelerating Dive	-	139.16 yd/sec-249.75 yd/sec
114 - 137	Pullout	4	To altitude of 958 yards
138 - 141	Starboard Turn	2	
142 - 145	Port Turn	2	
146 - 149	Port Turn	2	
150 - 157	Port Turn	3.5	Begin 10° Climb -
158 - 165	Starboard Turn	3.5	Velocity of 249.76
166 - 173	Port Turn	3.0	yd/sec is maintained
174 - 181	Starboard Turn	3.0	
182 - 193	Port Turn	3.0	
194 - 205	Starboard Turn	2.5	
206 - 221	Port Turn	2.5	
222 - 233	Starboard Turn	2.0	

Note that the port turn at points 142 - 145 is followed by another similar turn from points 146 - 149. The need for the additional segment rather than creating just one turn from points 142 - 149 will become clear later in the discussion.

The geometric relationships used to translate the desired track into mathematical terms include both basic, intuitive approaches and some requiring direct application to a given target orientation. These general techniques will now be discussed; the implementation of the particular track segments in a FORTRAN program may be found in Appendix B.

The following sub-sections portray the various possible maneuvers and track orientations used in developing the track for the MCSP.

A.1 The initial conditions include the starting point anywhere in the coordinate system with the velocity vector parallel to an axis. Specified parameters for a turn can include angle of turn, time of execution, tangential velocity,

constant radial acceleration, and final translation with respect to starting point. (Examples are: Points 1 - 33, 138-141, 146 - 149)

The geometrical arrangement is shown in Figure 5, and the following pertinent definitions and equations apply: a is the radial acceleration, vel is the tangential velocity, and T is the time of maneuver; thus

$$a = \frac{32.2}{3} \text{ yds/sec}^2 \cdot G \text{ of turn} = vel^2/R, \quad (22)$$

$$C = (4h(2R-h))^{\frac{1}{2}}, S = R\theta = 2R\sin^{-1}C/2R,$$

$$T = S/2 \cdot vel. \quad (23)$$

Using the pertinent equations certain parameters of the turn can be specified while assuring that the other parameters are reasonable. This results in values for R , θ , and the number of track points in the segment and provides incrementally growing $\Delta\theta$'s. Then each new track point is given by

$$X_n = X_o \pm R \sin \Delta\theta \quad (24)$$

$$Y_n = Y_o \pm (R - R \cos \Delta\theta). \quad (25)$$

Note: The optional signs above are dependent upon the orientation of the maneuver on the axes.

A.2 The initial conditions include the starting point anywhere with the velocity vector at some angle, τ , with respect to an axis. The turn is in the plane of two of the axes with certain parameters known, such as those mentioned in the previous sub-section. Four types of turn are possible in this scenario:

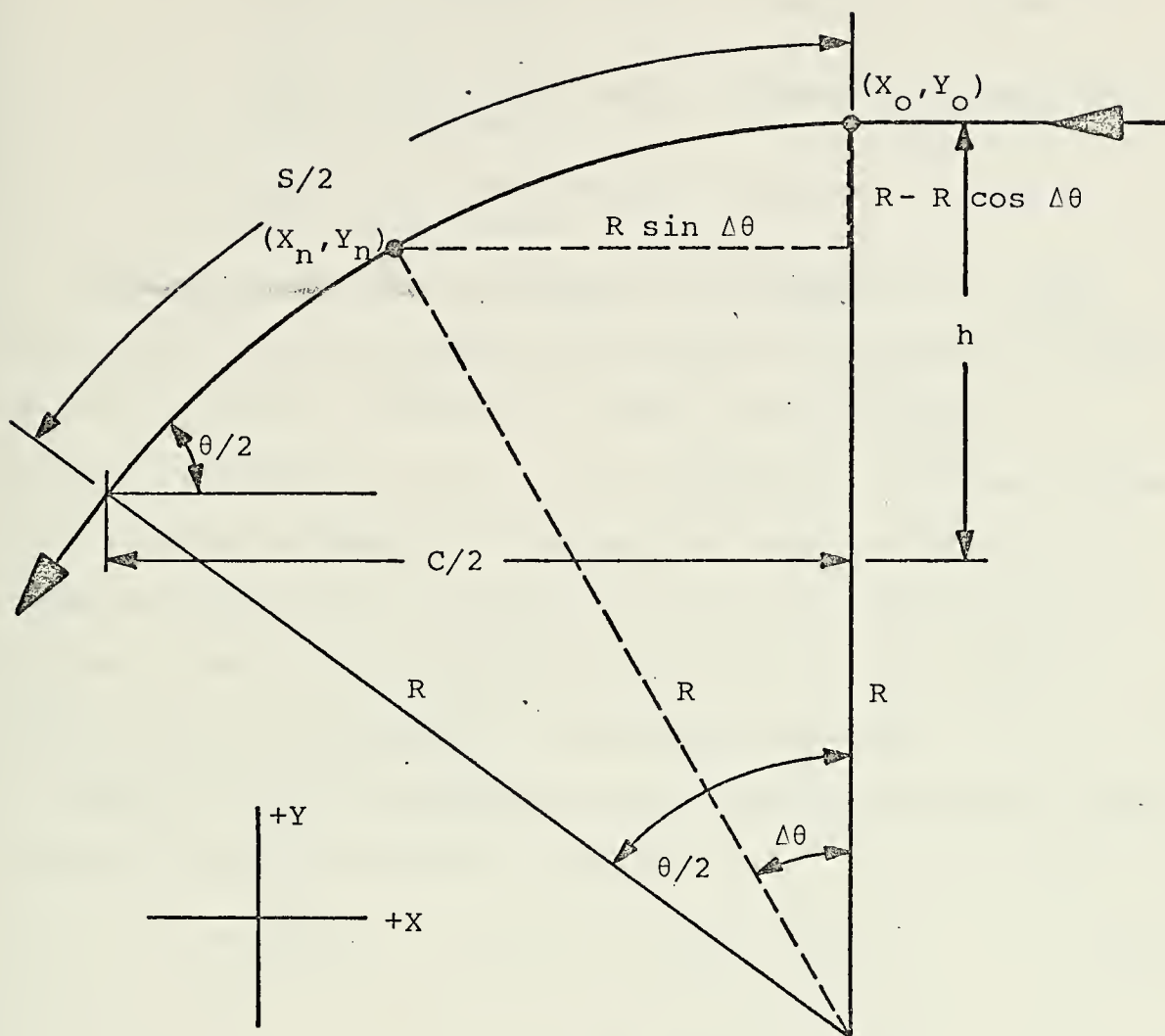


Figure 5. Geometry for Turn with Velocity Vector Initially Parallel to Axis.

- a - Initial velocity vector τ degrees to right of axis; right turn,
- b - Initial velocity vector τ degrees to right of axis; left turn,
- c - Initial velocity vector τ degrees to left of axis; right turn,
- d - Initial velocity vector τ degrees to left of axis; left turn.

The same pertinent equations as presented in A.1 are valid here, and the geometrical layouts are shown in Figures 6 and 7. Figure 6 applies to cases a and d above, and Figure 7 to cases b and c. For comparison all cases will be based on the assumption of an initial velocity vector at τ degrees with respect to the X axis and the turn in the XY plane. However, the procedure is directly applicable to other axes and planes with similar orientation.

The following development for obtaining the desired quantities $\Delta\phi$ and C' applies to Figure 6.

$$\alpha = \frac{180-\Delta\theta}{2}, \lambda = \beta - \alpha, \beta = 180 - (\tau + \Delta\theta) \quad (26)$$

$$\therefore \lambda = 180 - \tau - \Delta\theta - 90 + \frac{\Delta\theta}{2} = 90 - \tau - \frac{\Delta\theta}{2} \quad (27)$$

$$\text{and } \Delta\phi = 90 - \lambda = \tau + \frac{\Delta\theta}{2} \quad (28)$$

Also, it can be seen that $C' = 2 R \sin (\Delta\theta/2)$.

Similarly, the following development applies to Figure 7.

$$\alpha = \frac{180-\Delta\theta}{2}, \beta = 90 + (90-\tau) = 90 - \frac{\Delta\theta}{2} \quad (29)$$

$$\Delta\phi = \beta - \alpha = 180 - \tau - 90 + \frac{\Delta\theta}{2} = 90 - \tau + \frac{\Delta\theta}{2} \quad (30)$$

Again, it can be seen that $C' = 2 R \sin (\Delta\theta/2)$.

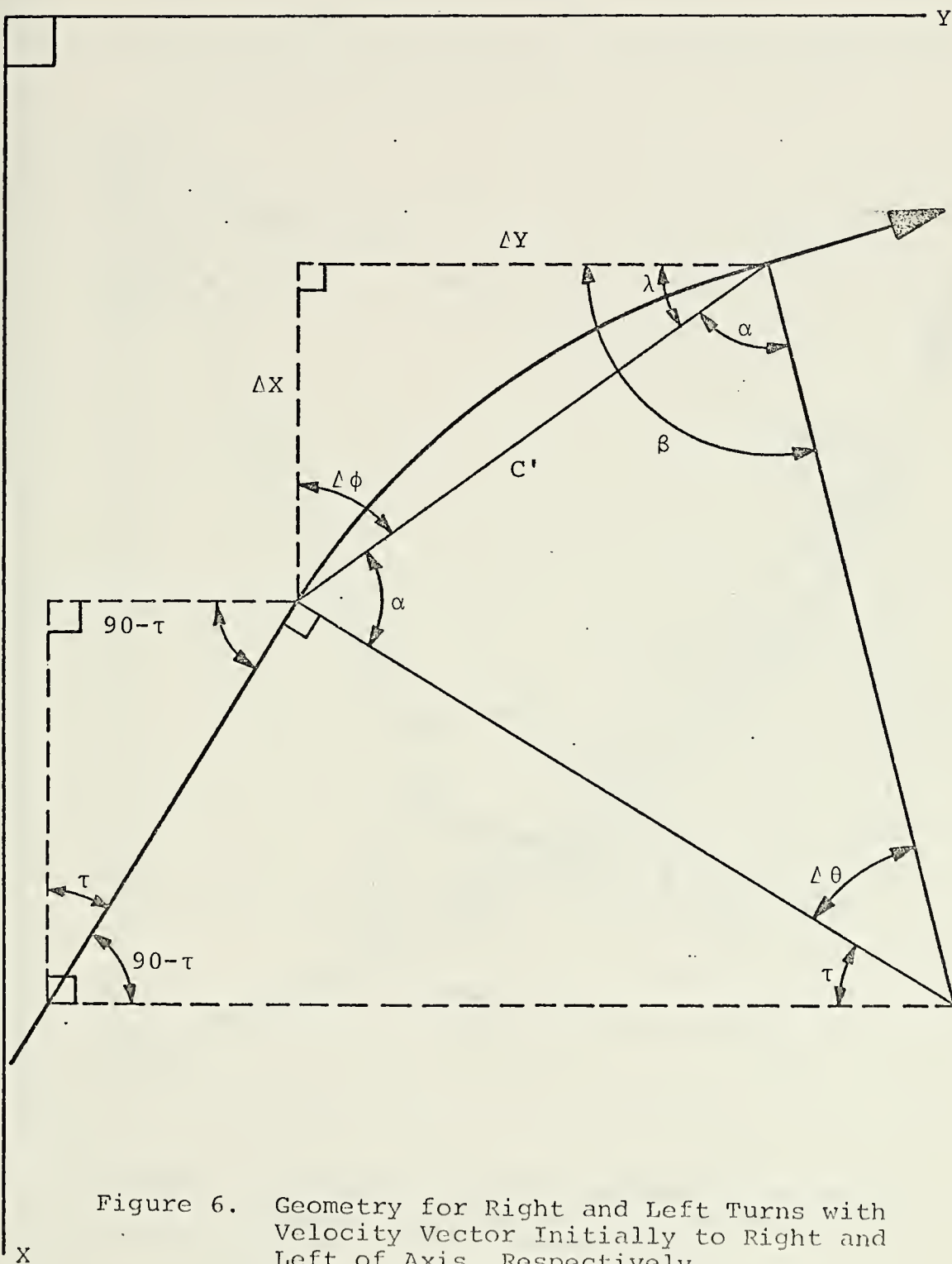


Figure 6. Geometry for Right and Left Turns with Velocity Vector Initially to Right and Left of Axis, Respectively.

Given that the quantities $\Delta\phi$ and C' are available (depending upon the choice of the variable parameters) the track points can be generated for the four cases. A summary for the four cases follows with the appropriate figures for a, b, c, and d included as Figures 8, 9, 10, and 11, respectively.

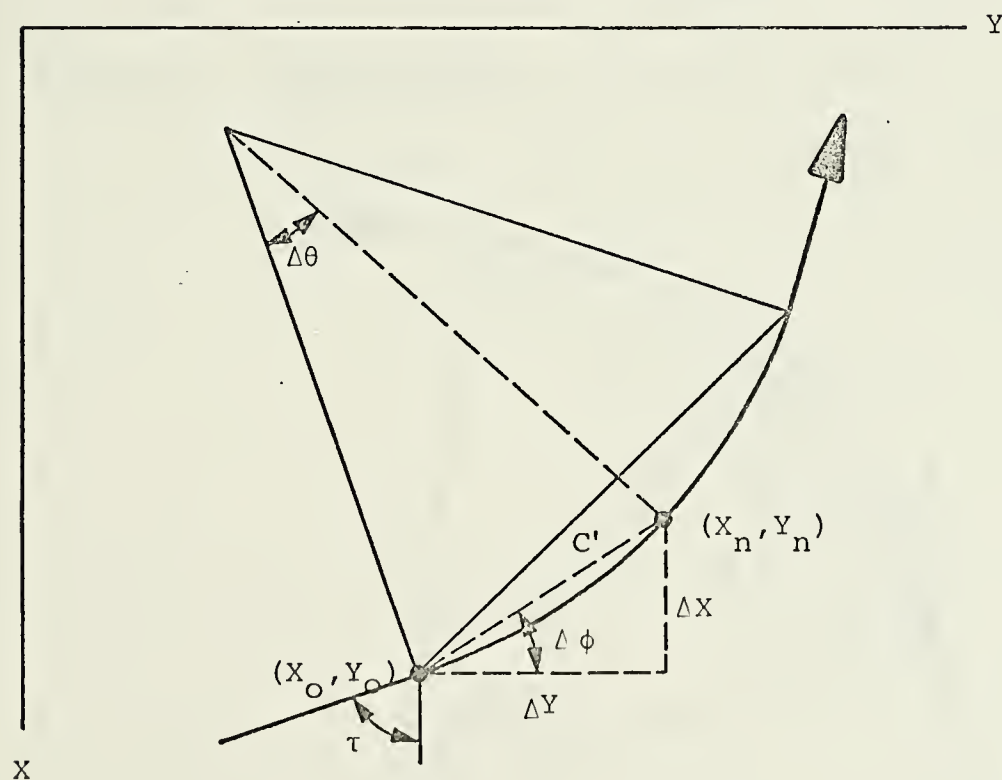
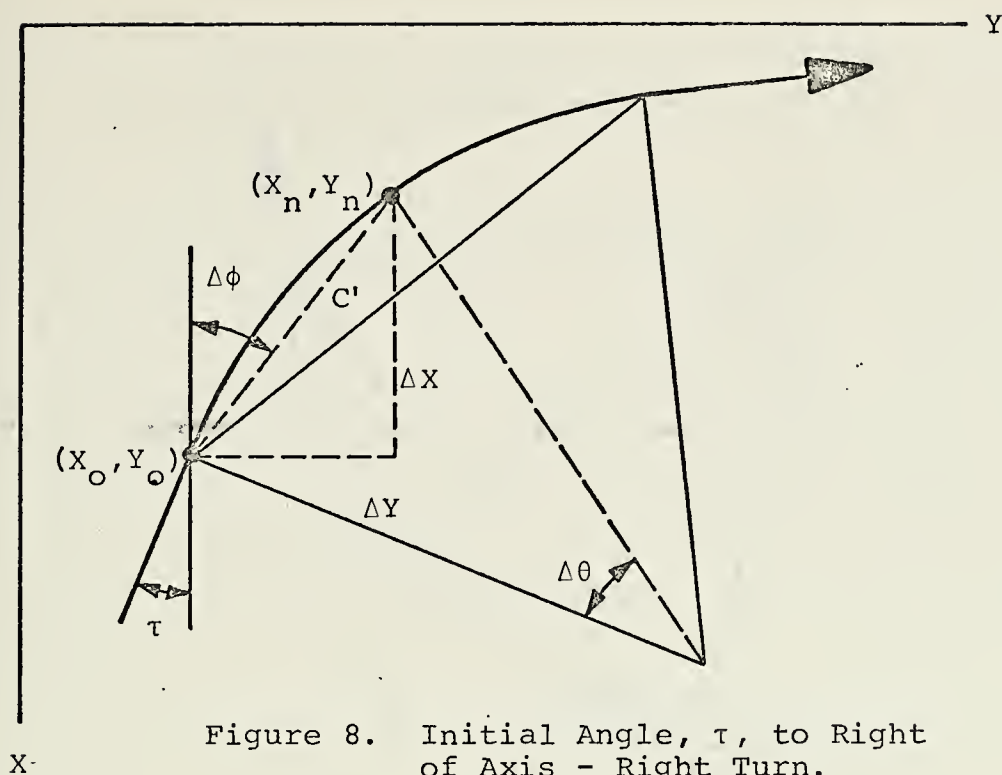
<u>A.2.a</u> τ to right - right turn (Example: Points 114-137)	$\Delta X = C' \cos \Delta\phi$ $\Delta Y = C' \sin \Delta\phi$ $X_n = X_o - \Delta X$ $Y_n = Y_o + \Delta Y$	(31)
--	--	------

<u>A.2.b</u> τ to right - left turn (Example: Points 142-145)	$\Delta X = C' \sin \Delta\phi$ $\Delta Y = C' \cos \Delta\phi$ $X_n = X_o - \Delta X$ $Y_n = Y_o + \Delta Y$	(32)
---	--	------

<u>A.2.c</u> τ to left - right turn	$\Delta X = C' \sin \Delta\phi$ $\Delta Y = C' \cos \Delta\phi$ $X_n = X_o - \Delta X$ $Y_n = Y_o - \Delta Y$	(33)
--	--	------

<u>A.2.d</u> τ to left - left turn	$\Delta X = C' \cos \Delta\phi$ $\Delta Y = C' \sin \Delta\phi$ $X_n = X_o - \Delta X$ $Y_n = Y_o - \Delta Y$	(34)
---	--	------

It should be noted that in using this approach it is assumed that when a turn is initiated at some angle, τ , with respect to an axis the quantity $(\tau + \theta)$ is less than or equal to 90 degrees. Otherwise, one must assure that the quantities ΔX and ΔY are added to or subtracted from the initial coordinates correctly. That is, beyond 90 degrees the X or Y



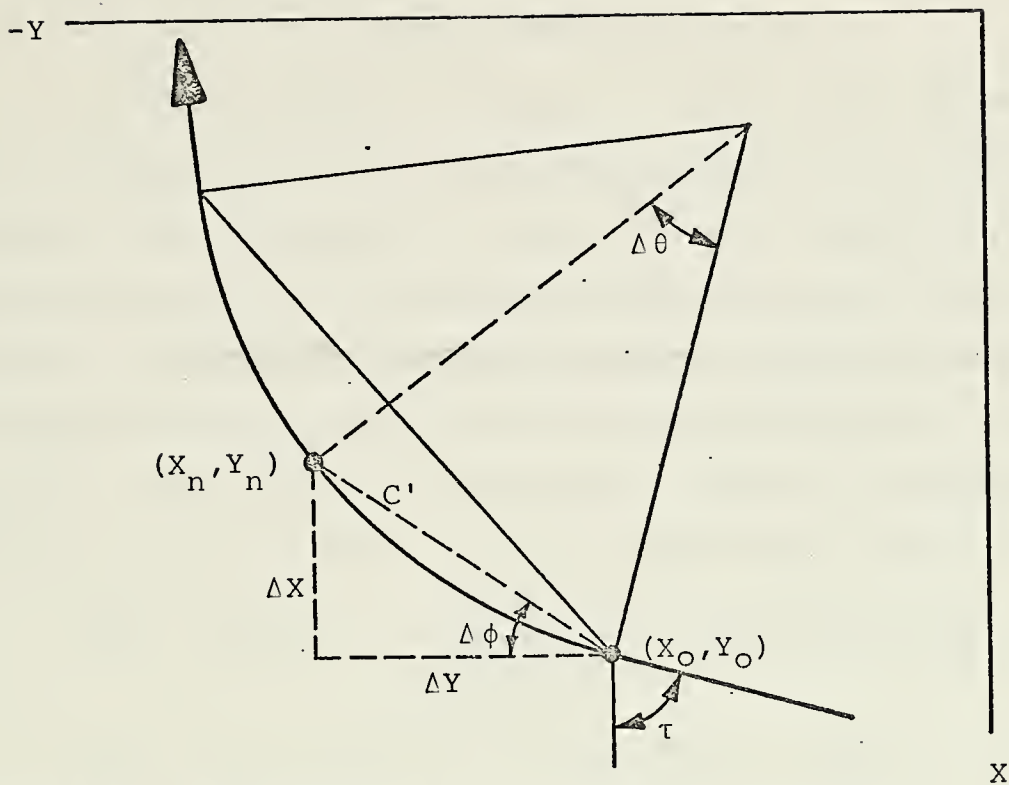


Figure 10. Initial Angle, τ , to Left of Axis - Right Turn.

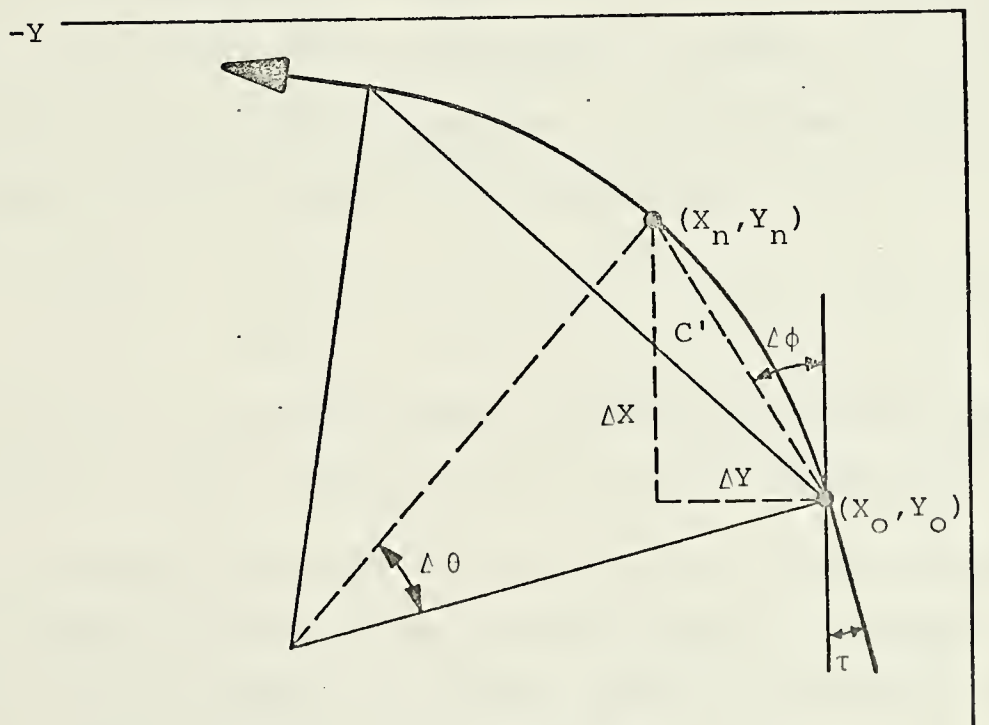


Figure 11. Initial Angle, τ , to Left of Axis - Left Turn.

coordinates may begin to change in directions opposite to that discussed in the scenario described above.

A.3 Initial conditions include a starting point anywhere in the coordinate system with constant-acceleration motion in a straight line. (Example: Points 34 - 113) Specified parameters may include initial and final velocity, time of execution, acceleration (assumed constant), angle of segment with respect to some plane, and final translation from starting point. The geometric arrangement is shown in Figure 12, and the pertinent kinematic relationships follow below

$$\text{Final Velocity} = \frac{2 \cdot S}{\left(\begin{array}{c} \text{Time of} \\ \text{Execution} \end{array} \right)} - \text{Initial Velocity}, \quad (35)$$

$$\text{Final Vel.} = \text{Init. Vel.} + \text{Acceleration} \cdot \text{Time}, \quad (36)$$

$$Z \text{ Translation} = S \cdot \sin \alpha, \quad (37)$$

$$X \text{ Translation} = S \cdot \cos \alpha, \quad (38)$$

$$S = \frac{1}{2} (\text{Init. Vel.} + \text{Final Vel.}) \cdot \text{Time}. \quad (39)$$

Using these equations to define a track segment, the incremental displacements can be defined as:

$$\Delta X = \Delta S \cdot \cos \alpha, \quad (40)$$

$$\Delta Z = \Delta S \cdot \sin \alpha. \quad (41)$$

Then each point along the track is defined by applying these incremental changes relative to the previous track point.

A.4 The last situation to be discussed is that applying to the evasive retreat of the aircraft from the seaborne target. In this scenario a gradual climb is initiated after passing over the ship, and the climb angle of β degrees

requires an additional perspective into the aircraft's motion. Using the techniques discussed previously in subsection A.2 the aircraft's track was defined in the XY plane of the coordinate system. If, however, the incremental Z motion is such as to give a β climb angle the apparent effect in the plane of the aircraft is to cause an increasing radially accelerating turn while maintaining the same constant-acceleration turn in the XY plane. It can be seen in Figure 13 that in the plane of the aircraft the radius of curvature decreases from R_1 to R_2 ; therefore, the radial acceleration increases as it is inversely related to the radius in a constant velocity model by $R = \text{Vel}^2 / \text{Acceleration}$. This apparent complication is justifiable in the context of actual operation in that an aircraft might, in fact, execute a turn with an increasing amount of radial acceleration. This causes no problems in the simulation as long as the angle, τ , between the appropriate axis and the aircraft's velocity vector is continuously updated in order to correctly initiate the succeeding maneuver.

Two plots of the aircraft motion throughout the entire maneuver are included as Figures 14 and 15. Figure 14 depicts the XZ plane, and Figure 15 shows a top view of the XY plane.

B. TRACK OPTIONS

The groundwork for the track development having been discussed, next follows a description of the options available in using this or any other track in the MCSP. The discussion

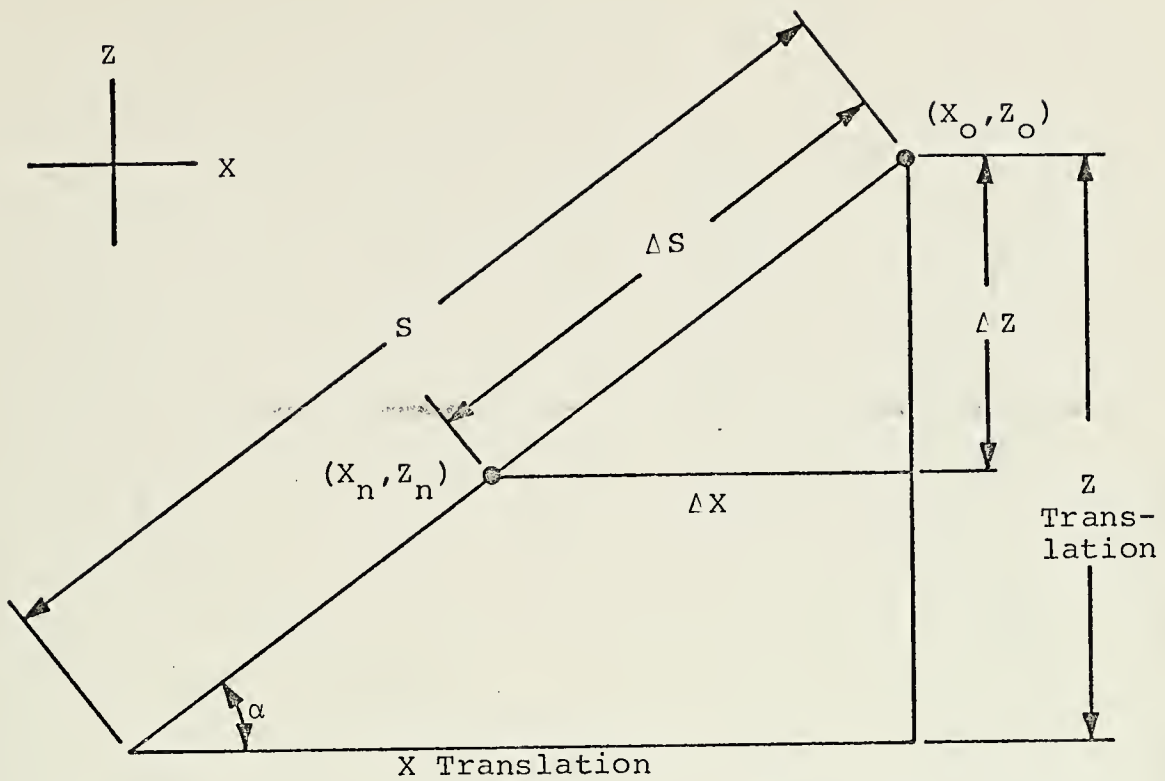


Figure 12. Constant Acceleration Motion in a Straight Line.

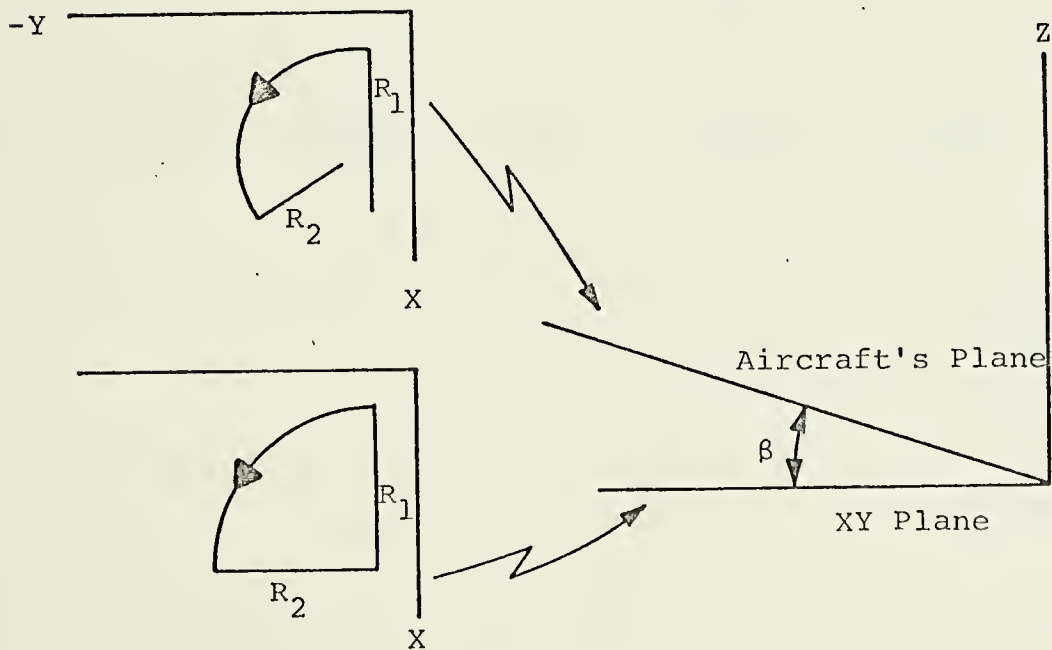
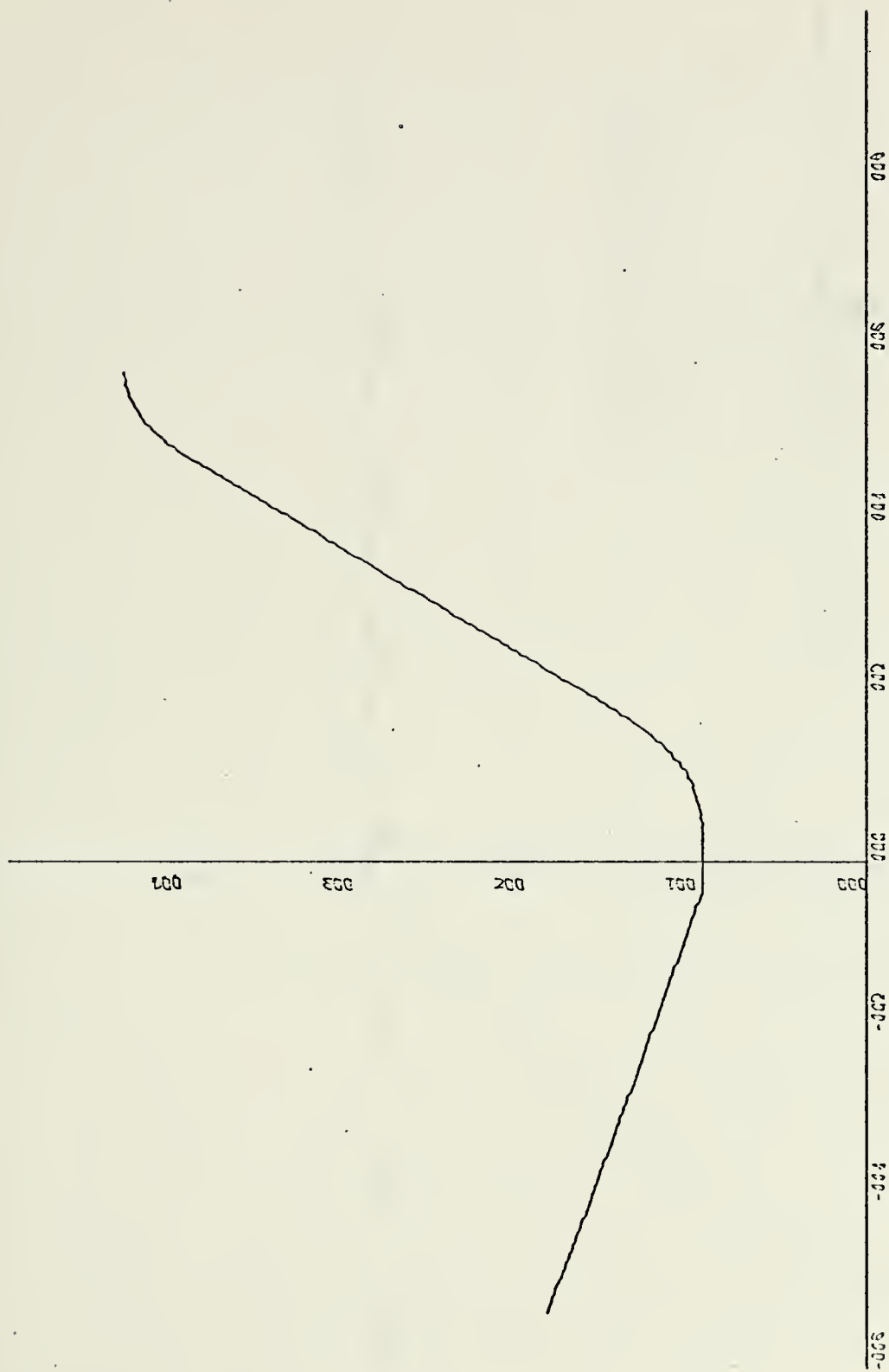


Figure 13. Constant Radial Acceleration Turn with Climb Angle of β .



X scale: 2000 yards/in
Z scale: 1000 yards/in

Figure 14. Aircraft Motion in XZ Plane.

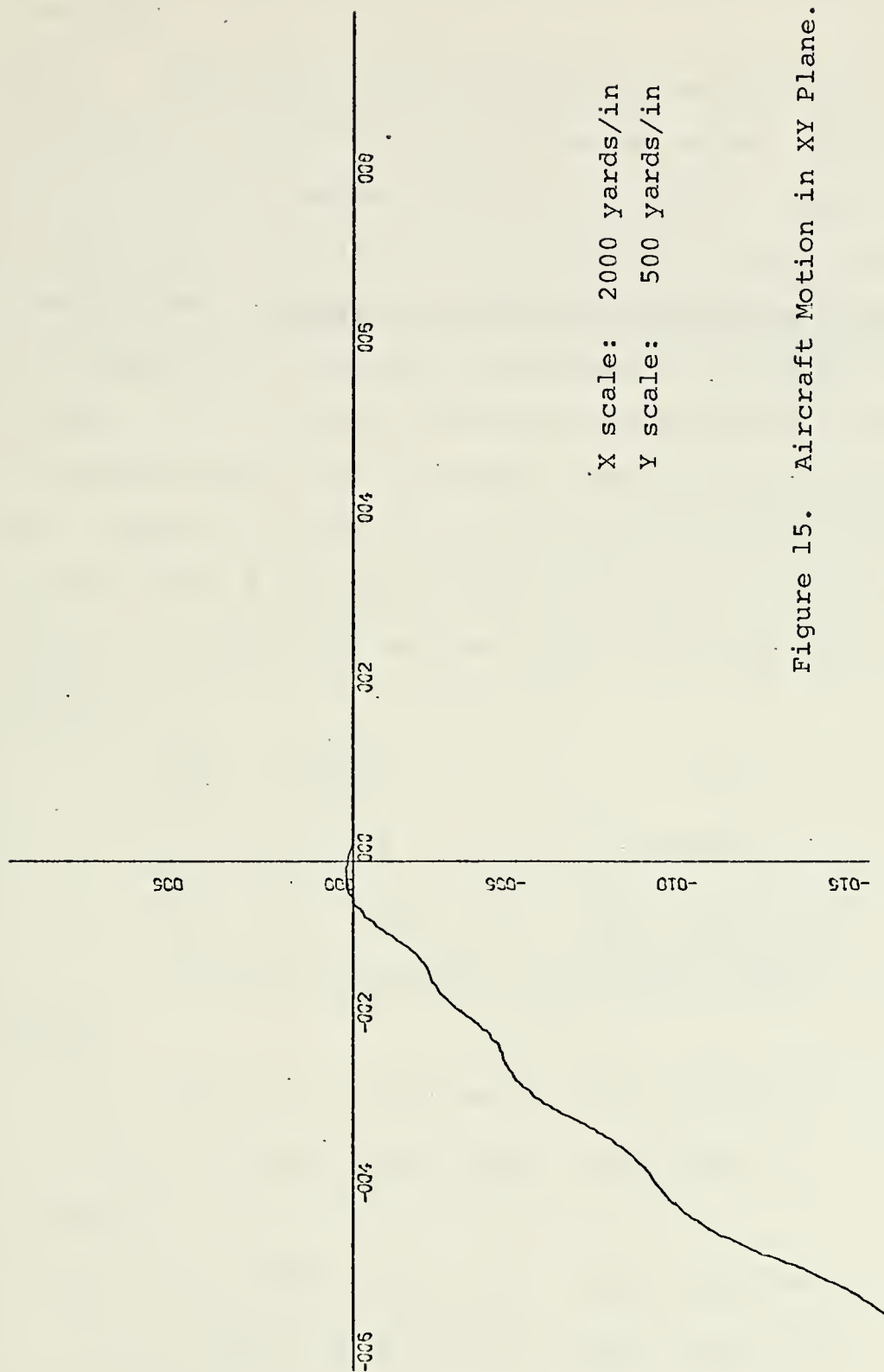


Figure 15. Aircraft Motion in XY Plane.

of implementing the desired options follows in a later section.

B.1 The first option available is the selection of direction of approach of the airborne target relative to the coordinate system centered at the ship. This rotational capability is achieved by inputting the original track in XYZ coordinates, converting into RBE, and adding a desired radial increment to the bearing coordinate of every point shown in Figure 16. If track velocities are also input to the program (an option that is covered later), it can be seen that the component in the Z direction is not affected while those in the X and Y directions are. Following is a derivation to handle this problem, and the vector presentation is shown in Figure 17.

$$|\text{VEL}'| = |\text{VEL}| \quad (42)$$

$$\dot{X} = -\text{VEL} \cdot \cos \beta \quad \dot{Y} = \text{VEL} \cdot \sin \beta \quad (43)$$

$$\dot{X}' = -\text{VEL}' \cdot \cos (\beta - \Delta) = -\text{VEL} \cdot \cos (\beta - \Delta) \quad (44)$$

$$\dot{Y}' = \text{VEL}' \cdot \sin (\beta - \Delta) = \text{VEL} \cdot \sin (\beta - \Delta) \quad (45)$$

Using

$$\cos (\beta - \Delta) = \cos \beta \cos \Delta + \sin \beta \sin \Delta \quad (46)$$

$$\sin (\beta - \Delta) = \sin \beta \cos \Delta - \cos \beta \sin \Delta \quad (47)$$

We have

$$\dot{X}' = -\text{VEL} \cdot \cos \beta \cos \Delta - \text{VEL} \cdot \sin \beta \sin \Delta \quad (48)$$

$$\dot{Y}' = \text{VEL} \cdot \sin \beta \cos \Delta - \text{VEL} \cdot \cos \beta \sin \Delta \quad (49)$$

And finally,

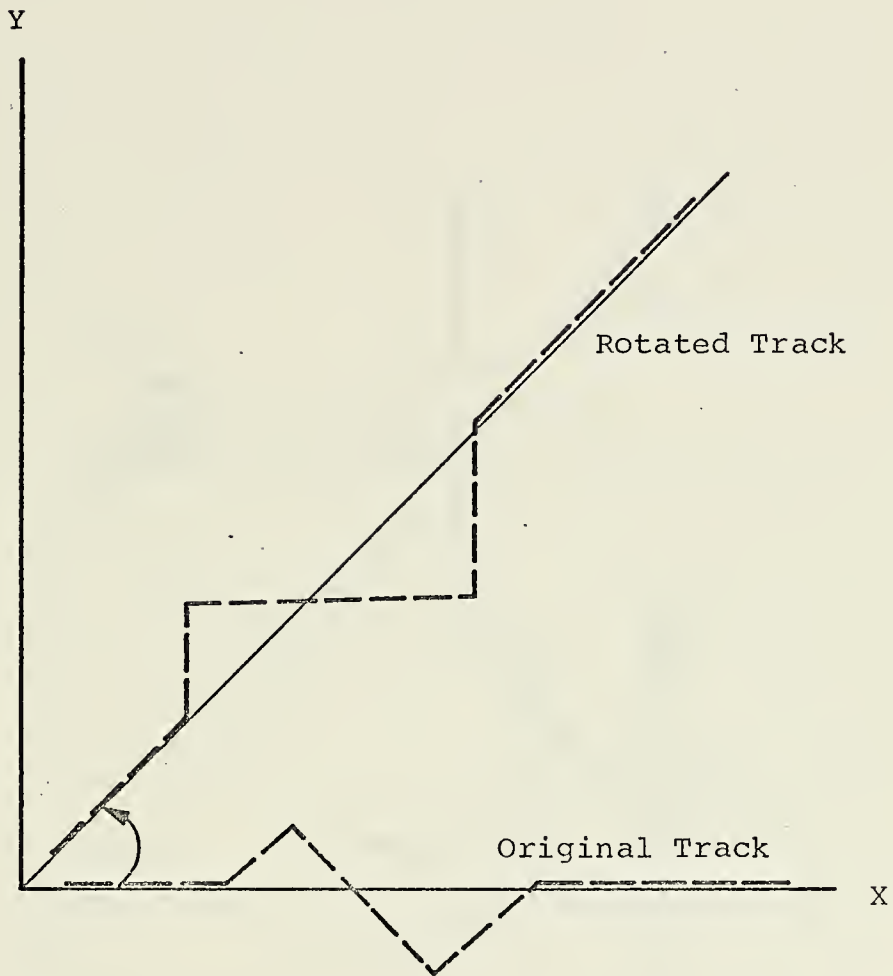


Figure 16. Rotation of Track.

Y

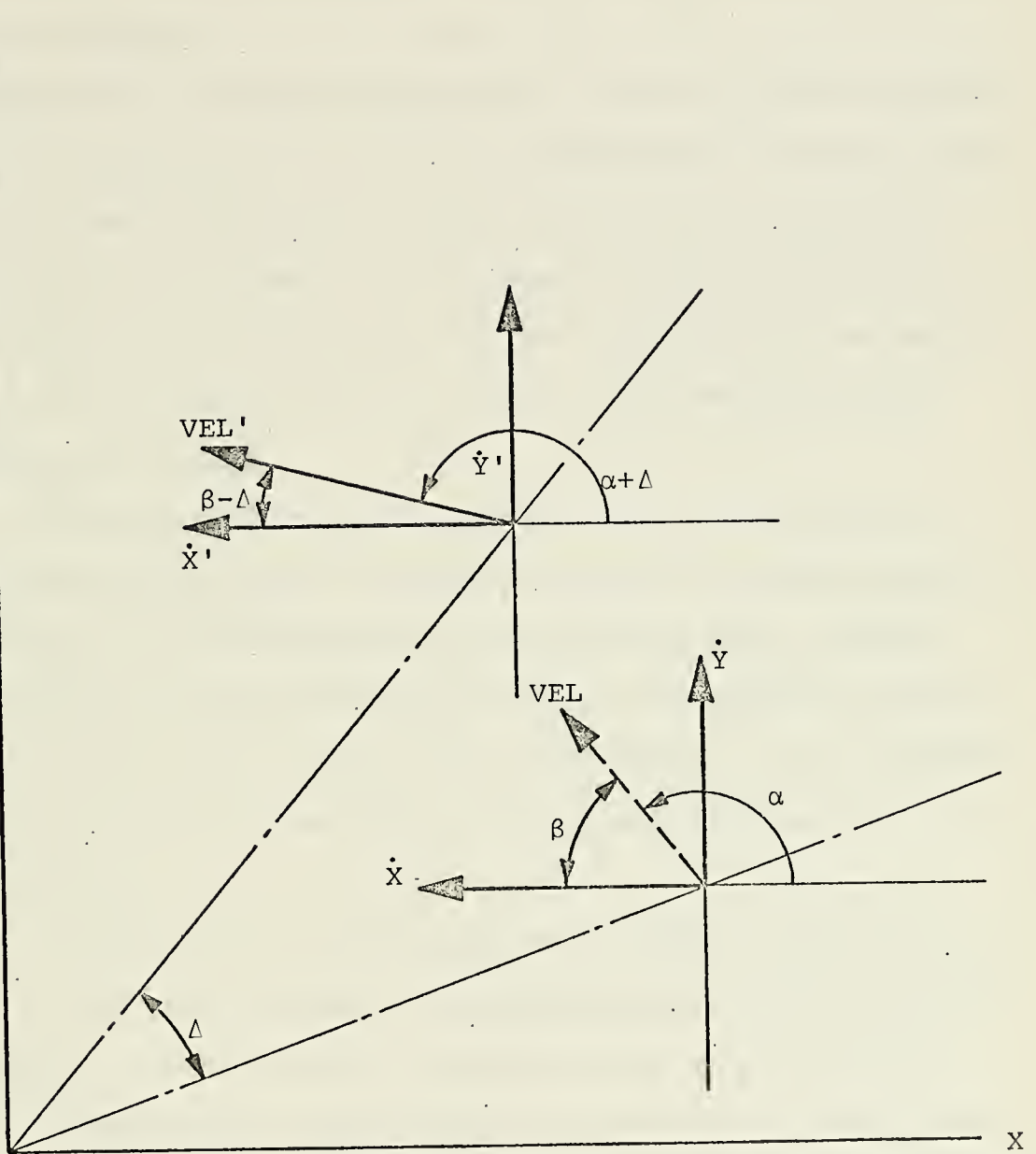


Figure 17. Velocity Vectors for Track Rotation.

$$\dot{X}' = \dot{X} \cos \Delta - \dot{Y} \sin \Delta \quad (50)$$

$$\dot{Y}' = \dot{Y} \cos \Delta + \dot{X} \sin \Delta \quad (51)$$

where \dot{X} and \dot{Y} are the original track velocities and Δ is the angle of rotation.

The MCSP is arranged to provide a choice of the following angles of approach: 0° , 45° , 90° , and 202.5° . If other than these choices is desired the MCSP can be easily altered by referring to the segment of the program several statements after label 13 in the listing of the MCSP included as Appendix C. The variable DEL can be set to the desired radial increment in radians.

B.2 The second option available is the translation of all of the track points by desired amounts for each coordinate axis. The implementation is discussed later, but the method simply involves adding the desired increments to all of the original track points' XYZ coordinates. The velocity components at each track point are not affected by the translation.

B.3 A third option is the choice of adding measurement noise to the track points. If requested, the noise is added to the RBE coordinates and conversion back into noisy XYZ occurs. This noise, as now exists in the MCSP is mean zero, and the user selects the desired standard deviations in range, bearing, and elevation (units are yards, radians, radians).

In conclusion, the procedure for implementation of a track and possible options requires that the XYZ data points be generated offline and read into the MCSP where various

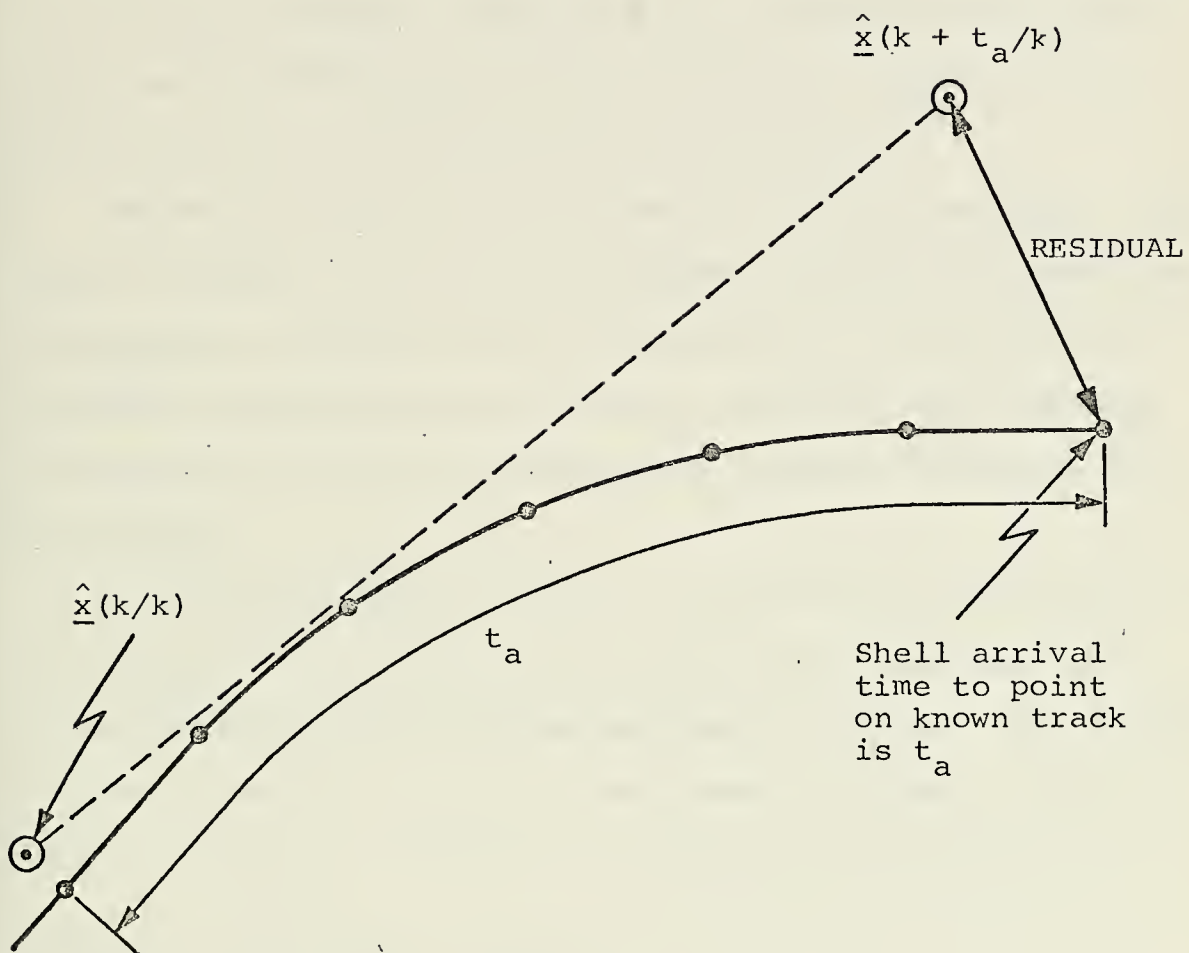
input parameters control the rotation, translation, and noise options applied.

C. SHELL-AT-TARGET ACCURACY

A feature which is an option in the MCSP provides approximate shell-at-target miss distances for additional filter evaluation. This feature is based on the firing table for a 5-inch 54-caliber gun and provides the approximate miss distances, or residuals, in X, Y, and Z directions and, also, the absolute miss distance with time of shell flight. Sections of the MCSP that are specifically responsible for these calculations are Subroutines ACCUR and STUDY.

The principles involved in obtaining the residuals will now be described. The first step required for implementation of the routine requires that the input XYZ track data be provided to program GUNFIRE, included as Appendix D, for offline calculation of the shell flight times to each point using interpolation in the 5-inch 54 firing table, included also in Appendix D. Program GUNFIRE is set up to provide these times for data points out to a range of 7500 yards and having an elevation of from 15 to 90 degrees above the horizon.

The known information includes the XYZ data points, the data point spacing, the shell times to each data point, and the ϕ , or state transition matrix, and the objective, as depicted in Figure 18, is to step back in time from a track point for which the shell arrival time, t_a , is known to a filter estimated point calculated an amount t_a earlier.



\bullet - True track points

\odot - Filter outputs

Figure 18. Shell-at-target Accuracy Diagram.

Knowing this estimated point $\hat{\underline{x}}(k/k)$ and the $\underline{\phi}$ matrix the filter's prediction of target location an amount t_a later is included in the vector defined by $\underline{\phi}^{(\frac{t_a}{T})} \hat{\underline{x}}(k/k)$ where T is the time between samples. The residual is then defined to be equal to the difference between the position states in the vector $\hat{\underline{x}}(k+t_a/k)$ and the actual track point.

Two major problems arise, however, in this approach. The first is that if t_a is not an integer multiple of track time increments, a filter estimate of states is not available. Secondly, implementation of this method generally requires raising the $\underline{\phi}$ matrix to a fractional power--a situation to be avoided.

A reasonable approximation to the desired results above is to step back in time to the first filter estimate prior to the desired time of filter estimation vector. This amounts to some fraction, α , of the time between samples, and the required vector of estimates is given, approximately, by

$$\hat{\underline{x}}(k+\alpha/k) = \underline{\phi}^{(\frac{\alpha}{T})} \hat{\underline{x}}(k/k). \quad \text{Hence,}$$

$$\hat{\underline{x}}(k+\alpha+t_a/k+\alpha) = \underline{\phi}^{(\frac{t_a}{T})} \hat{\underline{x}}(k+\alpha/k+\alpha) \quad (52)$$

$$\approx \underline{\phi}^{(\frac{t_a}{T})} \underline{\phi}^{(\frac{\alpha}{T})} \hat{\underline{x}}(k/k) \quad (53)$$

$$\approx \underline{\phi}^{(\frac{t_a+\alpha}{T})} \hat{\underline{x}}(k/k) \quad (54)$$

but $\frac{t_a+\alpha}{T}$ is an integer, thus solving the problem of raising $\underline{\phi}$ to a fractional power.

EXAMPLE: Given that the time for a shell to reach data point 100 is 9.7 time periods (the same as assuming a 1-Hz sampling rate) finds the filter's predicted position.

SOLUTION: Because $\hat{\underline{x}}(90.3/90.3)$ is not available, approximate it is $\underline{\phi}^{0.3} \hat{\underline{x}}(90/90)$. Then the filter's prediction vector at time 100 is given by

$$\begin{aligned}\hat{\underline{x}}(100/90) &= \underline{\phi}^{0.3} \underline{\phi}^{9.7} \hat{\underline{x}}(90/90) \\ &= \underline{\phi}^{10} \hat{\underline{x}}(90/90) .\end{aligned}\tag{55}$$

It should be noted that when the shell flight times are generated by GUNFIRE these are applicable only to the original track or a track rotated by the MCSP. If translation of a track is to be performed by the MCSP the shell-at-target miss distances generated will be erroneous, and this option should not be used.

D. ONLINE CALCULATION OF COVARIANCE MATRIX OF POSITION MEASUREMENT NOISE

The covariance matrix of measurement noise, \underline{R} , is a measure of the noise associated with the measurements and is used in calculating the gains. \underline{R} is defined as $E[\underline{n}(k) \underline{n}^T(k)]$ where $\underline{n}(k)$ is the additive noise vector at time k .

In problems where the measurements are linearly related to the filter states, \underline{R} can be approximated as a constant matrix for the entire track, an option available in the MCSP.

However, when the observations presented to the filter are range, bearing, and elevation (with noise assumed mutually independent) and the state equations are in rectangular coordinates, the required coordinate conversion generates a coupled, non-independent relationship among the noise terms

in the filtering coordinate system. This section presents a derivation to define the resultant \underline{R} matrix for the given coordinate conversion.

The radar measures range, r , bearing, θ , and elevation, ϕ , and these measurements are made available to the estimator at sample times with the parameter k as an index. The measurements also include independent noise terms $n_1(k)$, $n_2(k)$, and $n_3(k)$ that are assumed to be characterized by random, white distributions with mean zero and known variances. This implies that

$$E[n_i(k)]_{i=1,2,3} = 0 \quad \text{for all } k \quad (56)$$

$$E[n_1(k)n_1(j)] \quad \begin{cases} 0 & j \neq k \\ \sigma_R^2 & j = k \end{cases} \quad (57)$$

$$E[n_2(k)n_2(j)] \quad \begin{cases} 0 & j \neq k \\ \sigma_B^2 & j = k \end{cases} \quad (58)$$

$$E[n_3(k)n_3(j)] \quad \begin{cases} 0 & j \neq k \\ \sigma_E^2 & j = k \end{cases} \quad (59)$$

$$\begin{array}{l} E[n_1(k)n_2(j)] \\ E[n_1(k)n_3(j)] \\ E[n_2(k)n_3(j)] \end{array} \quad \begin{array}{l} 0 \\ 0 \\ 0 \end{array} \quad \left. \vphantom{\begin{array}{l} E[n_1(k)n_2(j)] \\ E[n_1(k)n_3(j)] \\ E[n_2(k)n_3(j)] \end{array}} \right\} \text{for all } k, j \quad (60)$$

The terms σ_R^2 , σ_B^2 , and σ_E^2 must be known. The coordinate system is the same as shown in Figure 2.

The rectangular coordinates of the target are

$$X(k) = r(k) \cos \theta(k) \cos \phi(k) \quad (61)$$

$$Y(k) = r(k) \sin \theta(k) \cos \phi(k) \quad (62)$$

$$Z(k) = r(k) \sin \phi(k) \quad (63)$$

However, the output of the radar is

$$S_1(k) = r(k) + n_1(k) \quad (64)$$

$$S_2(k) = \theta(k) + n_2(k) \quad (65)$$

$$S_3(k) = \phi(k) + n_3(k) \quad (66)$$

and what the estimator receives is given by the noisy XYZ values denoted by $Z_1(k)$, $Z_2(k)$, $Z_3(k)$ and defined as follows

$$Z_1(k) = S_1(k) \cos S_2(k) \cos S_3(k) \quad (67)$$

$$Z_2(k) = S_1(k) \sin S_2(k) \cos S_3(k) \quad (68)$$

$$Z_3(k) = S_1(k) \sin S_3(k) . \quad (69)$$

If equations (64) - (66) are substituted into equations (67) - (69), trigonometric identities used with subsequent expansion, and it is assumed that the bearing measurement noise, $n_2(k)$, and elevation measurement noise, $n_3(k)$, are small so that $\cos n_2(k) \approx 1$, $\sin n_2(k) \approx n_2(k)$, $\cos n_3(k) \approx 1$, and $\sin n_3(k) \approx n_3(k)$,³ we have

$$Z_1(k) = X(k) + v_1(k) \quad (70)$$

$$Z_2(k) = Y(k) + v_2(k) \quad (71)$$

$$Z_3(k) = Z(k) + v_3(k) \quad (72)$$

where $v_1(k)$, $v_2(k)$, and $v_3(k)$ represent additive noise and are given by

³In most radar tracking systems the standard deviations in bearing and elevation are less than .005 radians, or .286 degrees. The 3σ point, which includes 99% of this is, therefore, approximately 0.86 degrees. For comparison,
 $\sin 0.86^\circ = .0153$ vs. .015 radians
 $\cos 0.86^\circ = .9999$ vs. 1.

$$\begin{aligned}
v_1(k) = & - r(k) n_3(k) \cos \theta(k) \sin \phi(k) \\
& - r(k) n_2(k) \sin \theta(k) \cos \phi(k) \\
& + r(k) n_2(k) n_3(k) \sin \theta(k) \sin \phi(k) \\
& + n_1(k) \cos \theta(k) \cos \phi(k) \\
& - n_1(k) n_3(k) \cos \theta(k) \sin \phi(k) \\
& - n_1(k) n_2(k) \sin \theta(k) \cos \phi(k) \\
& + n_1(k) n_2(k) n_3(k) \sin \theta(k) \sin \phi(k) \quad (73)
\end{aligned}$$

$$\begin{aligned}
v_2(k) = & - r(k) n_3(k) \sin \theta(k) \sin \phi(k) \\
& + r(k) n_2(k) \cos \theta(k) \cos \phi(k) \\
& - r(k) n_2(k) n_3(k) \cos \theta(k) \sin \phi(k) \\
& + n_1(k) \sin \theta(k) \cos \phi(k) \\
& - n_1(k) n_3(k) \sin \theta(k) \sin \phi(k) \\
& + n_1(k) n_2(k) \cos \theta(k) \cos \phi(k) \\
& - n_1(k) n_2(k) n_3(k) \cos \theta(k) \sin \phi(k) \quad (74)
\end{aligned}$$

$$\begin{aligned}
v_3(k) = & r(k) n_3(k) \cos \phi(k) + n_1(k) \sin \phi(k) \\
& + n_1(k) n_3(k) \cos \phi(k) \quad (75)
\end{aligned}$$

Using the assumed mutual independence of the noise processes and the state variables which implies $E[n_i n_j] = E[n_i] \cdot E[n_j]$, $i \neq j$, gives

$$E[v_1(k)] = E[v_2(k)] = E[v_3(k)] = 0, \text{ for all } k, \text{ because}$$

$$E[n_1(k)] = E[n_2(k)] = E[n_3(k)] = 0.$$

Then, if it is assumed that $r^2(k) \gg \sigma_R^2$, the following terms of the R matrix can be defined. In every case

$$E[v_m(k) v_n(j)] = 0 \text{ for } j \neq k \text{ and } m, n = 1, 2, 3 \quad (76)$$

$$\begin{aligned} E[v_1(k) v_1(j)] &= r^2(k) \sigma_E^2 \cos^2 \theta(k) \sin^2 \phi(k) \\ &\quad + r^2(k) \sigma_B^2 \sin^2 \theta(k) \cos^2 \phi(k) \\ &\quad + r^2(k) \sigma_B^2 \sigma_E^2 \sin^2 \theta(k) \sin^2 \phi(k) \\ &\quad + \sigma_R^2 \cos^2 \theta(k) \cos^2 \phi(k) \end{aligned} \quad (77)$$

$$\begin{aligned} E[v_1(k) v_2(j)] &= E[v_2(k) v_1(j)] = \\ &\quad r^2(k) \cos \theta(k) \sin \theta(k) \sin^2 \phi(k) \\ &\quad \cdot [\sigma_E^2 - \sigma_B^2 \sigma_E^2] \\ &\quad + \sin \theta(k) \cos \theta(k) \cos^2 \phi(k) [-r^2(k) \sigma_B^2 + \sigma_R^2] \end{aligned} \quad (78)$$

$$\begin{aligned} E[v_1(k) v_3(j)] &= E[v_3(k) v_1(j)] = \\ &\quad \cos \phi(k) \sin \phi(k) \cos \theta(k) [\sigma_R^2 - r^2(k) \sigma_E^2] \end{aligned} \quad (79)$$

$$\begin{aligned} E[v_2(k) v_2(j)] &= r^2(k) \sigma_E^2 \sin^2 \theta(k) \sin^2 \phi(k) \\ &\quad + r^2(k) \sigma_B^2 \cos^2 \theta(k) \cos^2 \phi(k) \\ &\quad + r^2(k) \sigma_B^2 \sigma_E^2 \cos^2 \theta(k) \sin^2 \phi(k) \\ &\quad + \sigma_R^2 \sin^2 \theta(k) \cos^2 \phi(k) \end{aligned} \quad (80)$$

$$\begin{aligned} E[v_2(k) v_3(j)] &= E[v_3(k) v_2(j)] = \\ &\quad \sin \theta(k) \sin \phi(k) \cos \phi(k) [\sigma_R^2 - r^2(k) \sigma_E^2] \end{aligned} \quad (81)$$

$$E[v_3(k) v_3(j)] = r^2(k) \sigma_E^2 \cos^2 \phi(k) + \sigma_R^2 \sin^2 \phi(k) \quad (82)$$

A matrix summary of \underline{R} for the three-dimensional coordinate system is

$$R = \begin{bmatrix} E[v_1(k) v_1(j)] & E[v_1(k) v_2(j)] & E[v_1(k) v_3(j)] \\ E[v_2(k) v_1(j)] & E[v_2(k) v_2(j)] & E[v_2(k) v_3(j)] \\ E[v_3(k) v_1(j)] & E[v_3(k) v_2(j)] & E[v_3(k) v_3(j)] \end{bmatrix} \quad (83)$$

This covariance matrix is included in the MCSP by the subroutine RNOISE and functions in the following manner. The current filter estimate is used to predict ahead one time increment to define the expected position states. These predicted XYZ values are taken by routine RNOISE, converted to RBE, and, along with the standard deviations, σ_R , σ_B , and σ_E input as data, which, when squared, give the necessary variances, are used to define the covariance matrix of measurement noise required for calculating the gains for the next filter estimate. That is, $\hat{\underline{x}}(k + 1/k)$ is used to obtain \underline{R} for computing the gains needed to determine $\hat{\underline{x}}(k + 1/k + 1)$.

Again, this approach is an option which can be used with the Monte-Carlo simulation.

E. COVARIANCE MATRIX OF STATE EXCITATION

A filter designed to exactly follow a constant-velocity target will lag an accelerating target unless some technique

is incorporated to allow the filter to respond to the changes with increased gains to more heavily weight the latest observations. Increasing these gains is equivalent to increasing the bandwidth of the filter thus insuring that it is able to respond to deviations from the expected target performance.

The compromise made, however, is that additional measurement noise is able to enter into filter estimation and prediction through the increased bandwidth thus degrading the ability of the filter to reject measurement noise. There is, therefore, a balance which must be met between too small a covariance matrix of state excitation, \underline{Q} , and a resultant lagging filter and too large a \underline{Q} with subsequent erratic filter behavior.

The MCSP can accommodate the use of a \underline{Q} matrix in two ways--by using a constant input matrix of the anticipated covariance of state excitation in the calculation of all gains or by generating, online, \underline{Q} matrices through a procedure that is adaptive to the residuals detected between predicted and estimated filter states. The alternative to these methods, as the MCSP currently exists, is to use no \underline{Q} and input zeroes in its place.

The first of these methods requires some prior analysis of the target motion before the Monte-Carlo simulation is exercised, and with a modeled track, such as that designed for simulation with the MCSP, the variances can be evaluated using the track data.

The most direct method for obtaining terms to define a constant \underline{Q} matrix is to take successive differences of the position data to approximate velocities, accelerations, and acceleration rates. Having these arrays of accelerations and acceleration rates in each dimension of the coordinate system, the mean and variance of each array can be calculated with the usual statistical procedures. The resultant quantities provide the necessary variances for determining \underline{Q} for either constant-velocity or constant-acceleration models if the means produced are, in fact, close to zero (the assumed mean of random forcing). The implementation of this approach is included at the end of the track generating program in Appendix B, and the output of that program is also there showing the variances generated by the above technique.

While these variances are defined from actual track data it must be remembered that the resulting constant \underline{Q} matrix will inevitably be sub-optimal since the accelerations or acceleration rates do not, in most cases, occur uniformly over the track; for some points \underline{Q} will be "too small," thus not compensating for target motion and for other points \underline{Q} will be "too large" allowing for measurement noise degradation of filter performance.

The use of a well-chosen constant \underline{Q} matrix does, however, offer two distinct advantages. The first is that, if well chosen, it cannot help but improve the performance of a filter not dynamically adequate to follow a maneuvering target and, second, it offers simplicity and subsequent reduced

computation time and core area requirements in comparison to an adaptive technique.

The \underline{Q} matrix, when implemented as a constant matrix, is defined by

$$\underline{Q} = \underline{\Gamma} E[\underline{w} \underline{w}^T] \underline{\Gamma}^T \quad (84)$$

where $E[\underline{w} \underline{w}^T]$ is the covariance matrix of state excitation, and $\underline{\Gamma}$ is necessary to relate changes in velocity or acceleration for the constant-velocity or constant-acceleration filters, respectively, through successive integrations, to the appropriate states. For the XYZ coordinate system, when the random forcing is assumed to be de-coupled,

$$E[\underline{w} \underline{w}^T] = \begin{bmatrix} \sigma_X^2 & 0 & 0 \\ 0 & \sigma_Y^2 & 0 \\ 0 & 0 & \sigma_Z^2 \end{bmatrix}$$

The matrix $\underline{\Gamma}$ for a de-coupled constant-velocity filter in three dimensions is given by

$$\underline{\Gamma}_V = \begin{bmatrix} \frac{T^2}{2} & 0 & 0 \\ T & 0 & 0 \\ 0 & \frac{T^2}{2} & 0 \\ 0 & T & 0 \\ 0 & 0 & \frac{T^2}{2} \\ 0 & 0 & T \end{bmatrix}$$

where T is the time between measurements. The constant-acceleration $\underline{\Gamma}$ matrix for a de-coupled target motion model is

$$\underline{\Gamma}_A = \begin{bmatrix} T^3/6 & 0 & 0 \\ T^2/2 & 0 & 0 \\ T & 0 & 0 \\ 0 & T^3/6 & 0 \\ 0 & T^2/2 & 0 \\ 0 & T & 0 \\ 0 & 0 & T^3/6 \\ 0 & 0 & T^2/2 \\ 0 & 0 & T \end{bmatrix}$$

In the MCSP the matrices $\underline{\Gamma}$ and $E[\underline{w} \underline{w}^T]$ are input separately, and the necessary matrix products are formed to define the \underline{Q} matrix.

The second of the two methods available is an adaptive technique described in (4). The \underline{Q} matrix, at any point along the track, is defined to be

$$\begin{aligned} \underline{Q}(k) = & K_1 [\hat{\underline{x}}(k/k) - \hat{\underline{x}}(k/k-1)] [\hat{\underline{x}}(k/k) \\ & - \hat{\underline{x}}(k/k-1)]^T - K_2 \cdot \underline{Q}(k-1) \end{aligned} \quad (88)$$

where $\hat{\underline{x}}(k/k)$ is the filter's state estimation vector at time k and $\hat{\underline{x}}(k/k-1)$ is the state prediction vector determined at time $k-1$. It is this residual between the predicted states at time k using knowledge through time $k-1$ and the estimated

states at time k which offers a measure of the state excitation acting to cause the discrepancy. The terms K_1 and K_2 are weighting constants used to limit the degree to which \underline{Q} can change from one point to the next. By increasing K_1 , \underline{Q} becomes much more responsive to residuals and, also, noisy estimates while an increase in K_2 results in a type of damping on the \underline{Q} matrix.

This is, to say the least, a seat-of-the-pants approach, but, given a very complex modeling scenario, it offers many points in its favor. First, it is very simple and easy to implement with small additional computation time and storage required. Secondly, it does, in fact, vary the magnitude of \underline{Q} quickly, from point to point, rather than waiting for some threshold to be passed before alteration. Also, in a steady-state situation where the residual goes to zero, so does \underline{Q} , thus maintaining minimum bandwidth as desired.

It should be noted that the nature of gain calculations requires that the \underline{Q} matrix generated at point k is used to determine the gains for calculating the filter estimates at time $k + 1$, but given a reasonably high sampling rate the loss in performance should not be significant.

This adaptive method is easily requested, as will be shown in the section on using the MCSP, and the logic for its implementation appears at the end of subroutine UPDATE.

F. INITIALIZATION

The initialization of a Kalman filter can greatly affect the speed with which filter estimates and predictions improve.

Many techniques are available, and three options exist in the MCSP for user selection.

The first is a default option which is used if the other two methods are not requested. The filter initializing vector, $\hat{x}(0/-1)$, has its position states defined as the first observation, z , and the remaining states are initialized to zero. In conjunction with this choice it is advisable to initialize the prediction covariance matrix with very large values to assure large gains during the transient period.

The second possibility is to read in the initial conditions, an option that will be described in the next section. This is fine if reasonable initial conditions are known, but care must be taken not to detract from filter performance by choosing erroneous values offering worse initial conditions than the default case. Again, it pays to choose a large prediction covariance matrix if the gains are not precalculated.

The third option is to have the MCSP generate "synthetic" estimates by solving difference equations, online. With this method it takes two measurements to generate a velocity estimate and a third to generate an acceleration estimate if needed. After either two or three measurements, depending on whether a constant-velocity or constant-acceleration model is used, an initializing estimation covariance matrix is generated as follows in the MCSP:

Constant-Velocity Model

$$\underline{P}(1/1) = \begin{bmatrix} 1 & 1/T & 0 & 0 & 0 & 0 \\ 1/T & 2/T^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1/T & 0 & 0 \\ 0 & 0 & 1/T & 2/T^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1/T \\ 0 & 0 & 0 & 0 & 1/T & 2/T^2 \end{bmatrix} \quad (89)$$

Constant-Acceleration Model

$$\underline{P}(2/2) = \begin{bmatrix} 1 & 1/T & 1/T^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/T & 2/T^2 & 3/T^3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/T^2 & 3/T^3 & 6/T^4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1/T & 1/T^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/T & 2/T^2 & 3/T^3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/T^2 & 3/T^3 & 6/T^4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1/T & 1/T^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/T & 2/T^2 & 3/T^3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/T^2 & 3/T^3 & 6/T^4 \end{bmatrix} \quad (90)$$

The appropriate estimation covariance matrix is then used to generate a prediction covariance matrix for calculating the next gain matrix. With this option it is not necessary to input an initial prediction covariance matrix. The logic for its implementation can be found in subroutine UPDATE of the MCSP included as Appendix C.

IV. DESCRIPTION OF MONTE-CARLO SIMULATION PROGRAM (MCSP)

A. APPLICATION

The Monte-Carlo Simulation Program is designed for use in comparing 6th and 9th-order filter evaluations of airborne tracking methods and incorporates user controlled selection of the options mentioned in the previous sections and several other features that are discussed in this section. A listing of the program in FORTRAN IV appears in Appendix C. Comments appear liberally throughout the listing to aid in following the flow of the program as well as for pertinent information related to variable definitions and input instructions. The following sub-sections elaborate on the procedures to follow in using the MCSP and describe the output provided.

B. INPUTS

The input cards are described in sequence as required by the MCSP with description of the different arrangements dictated by the options requested. Unless otherwise specified the data should be double precision, D, format and right justified in the fields designated because calculations in the MCSP are done in double precision to minimize truncation errors.

<u>CARD 1</u>			
COL(S)	VARIABLE (ARRAY) NAME	FORMAT	DESCRIPTION
1 - 10	LOOP	I 10	Integer specifying the number of Monte-Carlo runs desired
11 - 20	KVEL	I 10	An integer "1" specifies that true velocity data for the track is available and is to be input

COL(S)	VARIABLE (ARRAY) NAME	FORMAT	DESCRIPTION
31 - 44	LOOK	14 I 1	<p>A 14-element integer array for outputting plots on the high speed printer. A "1" in the specified column produces a plot as follows:</p> <p>31 - Mean miss distance when shell arrives (from shell-at-target calculations). Default option automatically prohibits this output if the gunfire option is not requested.</p> <p>32 - Mean filter estimation error in X position calculation</p> <p>33 - Same for Y coordinate</p> <p>34 - Same for Z coordinate</p> <p>35 - Variance of X position estimates</p> <p>36 - Same for Y</p> <p>37 - Same for Z</p> <p>38 - Mean filter estimation error in X velocity calculation</p> <p>39 - Same for Y</p> <p>40 - Same for Z</p> <p>41 - Variance of X velocity estimates</p> <p>42 - Same for Y</p> <p>43 - Same for Z</p> <p>44 - Estimated target tangential velocity in ft/sec</p>
51 - 70	TIME	D 20.0	Time between measurements in secs.

CARD 2

COL(S)	VARIABLE (ARRAY) NAME	FORMAT	DESCRIPTION
1 - 5	N	I 5	Integer specifying the number of states, i.e. order of filter such as 6 or 9
6 - 10	NN	I 5	An integer specifying the number of track points or measurements

COL(S)	VARIABLE (ARRAY) NAME	FORMAT	DESCRIPTION
11 - 15	OPT	I 5	An integer "1" causes the program to generate synthetic measurements for filter initialization
16 - 20	BEGIN	I 5	An integer "1" signifies that the filter initialization vector $\hat{x}(0/-1)$ is to be input data
21 - 25	IR	I 5	An integer "0" causes the program to generate, online, R matrices as discussed in III.D. An integer "1" signifies that a constant R matrix is input data.
26 - 30	ADAPTQ	I 5	An integer "1" incorporates the adaptive Q matrix calculations
31 - 35	IG	I 5	An integer "1" causes the MCSP to punch out the gain matrices for each point on the track with one row of a matrix per card and having two coded integers in the first 10 columns of each card. The first specifies the track point and the second gives the row of the gain matrix for that point on the track. The gains are punched in a D format in columns 11-30, 31-50, and 51-70 of each card. <u>NOTE: It is necessary to change the second control card of the source deck from // EXEC FORTCLG to // EXEC FORTCLGP when using this option.</u>
36 - 40	IGAIN	I 5	This integer parameter controls the handling of gains in the MCSP. An integer "0" causes gains to be generated online for each MC run. An integer "1" generates gains on the first run with a noiseless track and uses these gains for all subsequent noisy tracks. An integer "2" causes gains to be read in by rows, 3 number per card. The data should be in D format in 20 column fields beginning in column 11.
41 - 80	LABEL	10 A 4	A 10-element alphanumeric array for run identification that appears at beginning of output

CARD 3

COL(S)	VARIABLE (ARRAY) NAME	FORMAT	DESCRIPTION
1 - 15	RR	D 15.0	Standard deviation of noise added to range data in yards. In MCSP this is added to .001. Range to provide a realistic figure
16-30	RB	D 15.0	Standard deviation of noise added to bearing data in radians
31-45	RE	D 15.0	Standard deviation of noise added to elevation data in radians
46 - 60	K1	D 15.0	Weighting constant for adaptive <u>Q</u> option which multiplies previous <u>Q</u>
61 - 75	K2	D 15.0	Weighting constant for adaptive <u>Q</u> option which multiplies new residual matrix

CARD 4

COL(S)	VARIABLE (ARRAY) NAME	FORMAT	DESCRIPTION
1 - 5	IDIR	I 5	An integer which specifies the rotation of the target track desired: 0 - no rotation 1 - 45° counterclockwise 2 - 90° counterclockwise 3 - 200.5° counterclockwise
6 - 10	INDEX	I 5	An integer controlling the addition of measurement noise to the track 0 - no noise 1 - add noise
11 - 15	IGUN	I 5	An integer "1" causes MCSP to calculate and output shell-at-target residuals
16 - 35	TRANSX	D 20.0	Desired translation of all X-axis data points in yards
36 - 55	TRANSY	D 20.0	Same for Y axis
56 - 75	TRANSZ	D 20.0	Same for Z axis

Card 5 begins the input section for matrices, and, therefore, the description given is for a typical card of the matrix with an indication of the number of cards that should be in the matrix. This section also depends upon the options that have been chosen above, and the possible combinations are discussed below.

- 1 - If gains are to be read in ($IGAIN = 2$) they are done so here under the format specified under the $IGAIN = 2$ description. There should be $NN \cdot N$ cards. (Next go to #1 below to read ϕ matrix.)

The following optional matrices apply when gains are calculated by the MCSP.

- 2 - If the R matrix is to be computed by the MCSP and synthetic measurements for initialization are not requested ($OPT \neq 1$) the following sequence applies.
 - (a) Read initial prediction covariance matrix, $P(0/-1)$, an $N \times N$ matrix. Data is read with an 8D10.5 format, thus there are N cards for 6th-order and $2 \cdot N$ cards for 9th-order with the 9th element of each matrix row on a card by itself in the first 10 columns following the card with the first 8 elements punched.
 - (b) Read initial matrix of random forcing covariances, $E[\underline{w} \underline{w}^T]$, a (3×3) matrix with the same input format as (a).
 - (c) Read the $(3 \times N)$ \underline{f} matrix with the same format as (a).
- 3 - If R is to be a constant matrix, the instructions in 2 above apply with the addition of reading the (3×3) R matrix between (a) and (b). The format is the same as 2 (a) above.
- 4 - If gains are calculated online and synthetic measurements are desired for filter initialization ($OPT = 1$) it is not necessary to provide a prediction covariance matrix. Therefore, in this case either the matrix of expected forcing covariances or a covariance matrix of measurement noise should appear first in this section of optional matrices.

These optional matrices having been arranged correctly, the next two matrices follow for all four possibilities.

- 1 - The $N \times N$ plant dynamics matrix, $\underline{\phi}$, is read. Again, the format follows that described in case 2 (a) above.
- 2 - The $3 \times N$ (for a three-dimensional problem) measurement matrix, \underline{H} , is read next with the same format as for $\underline{\phi}$. There should be either six or 12 cards depending on the order of the filter.

The next major group of cards pertains to the track data. The format for all track data is to read the XYZ components for a given point with a 3D20.0 specification for fields beginning in column nine. If true velocities are available and to be read they should be punched according to the same format, and the velocity data card for a given track point should follow immediately after the corresponding position data card and prior to the next point's position data card. There should be either NN or $2 \cdot NN$ cards in this segment depending on whether velocity data is available or not.

Following the track data, if IGUN = 1 implying that the shell-at-target calculations are desired, should appear the shell flight times punched one per card in the first 20 columns. There should be NN of these cards, and the format statement controlling their input is D20.0.

The last two cards possibly needed are for the initial conditions for the state vector $\underline{\hat{x}}(0/-1)$, called for when BEGIN = 1. The same format is followed as for the track data with the position data on the first card and velocity data on the second. In the case of a 9th-order filter the acceleration initial conditions are automatically set to zero.

This concludes the discussion of input data organization, and a complete sample data set is included in an example in Appendix E.

C. OUTPUT

Most of the output provided by the MCSP is self-explanatory, but a summary is included here with some additional insight into what the output actually means and a few peculiarities that the user should be aware of. A complete sample output is included with the example in Appendix E.

The first general output segment includes specification of some of the input parameters as well as the initial condition matrices. The matrices printed depend on what is relevant to the run requested. For example, if gains are read in, no prediction covariance matrix or random forcing matrix is provided. Also, if the adaptive Q method is used the actual initial state excitation matrix is printed instead of the output for constant-Q filter runs which includes the variances of random forcing that are provided as input data.

The next output segment, if requested, is the performance table showing the shell-at-target residuals. Because it takes several seconds for a shell to reach the target after filtering has been initiated, the output shows no residuals until this flight time has been satisfied. In the case of the track designed for the MCSP with no track translation, this amounts to 36 time increments. Also printed is a figure of merit for the given filtering scheme which is the average shell-at-target miss distance for those track points where it is applicable.

Following the gunfire figure of merit is a filter performance factor representing the rms error in filter position estimates averaged over the entire track. If true velocity data is available, a similar performance factor is printed which represents the rms error in filter velocity estimates averaged over the entire track.

The succeeding output table displays actual filter estimation accuracy by showing the mean and variance of the error between the filter estimate and the true track data. If velocity data is not available, the output for the velocity statistics presents the mean and variance of the estimates themselves.

The last and optional set of output is the plots requested on the first data card. These are produced by the routine PLOTP with one plot per page and headings already provided for the three-dimensional XYZ coordinate system.

V. SUMMARY OF REPRESENTATIVE SIMULATION RUNS

A. DESCRIPTION OF RUNS AND NUMERICAL RESULTS

This section presents the results of the simulations using some of the various combinations of options available in the MCSP. Rather than include the extensive outputs of the simulations only relative figures of merit for the shell-at-target accuracy calculations and filter performance factors are listed in Table II to provide some insight into the effects of possible combinations of selected options. In addition, results are included that were obtained using the MK-86 filter simulation developed in [5].

Descriptions of the various runs follow that include mention of the unique options used for each run. Unless otherwise specified the controls and initial conditions used in all runs include:

- (a) 100 Monte-Carlo runs,
- (b) ϕ , Γ and H are the same as described previously for 6th and 9th-order filters,
- (c) $P(k/k-1)$ is "large" (1.D05 for all diagonal terms),
- (d) Covariances of random forcing input matrix = 0,
- (e) Standard deviations of Gaussian noise added to track are $\sigma_R = 5 + .001 \times \text{Range}$, $\sigma_B = .002$ radians, $\sigma_E = .002$ radians,
- (f) No rotation or translation of track data,
- (g) Initialization of filter by letting the first observation define the position states in $\hat{x}(0/-1)$.

Reference made to "Constant Q" in the descriptions signifies that the covariances of random forcing used for defining the constant Q matrix are those calculated by successive differences in the track generating program. It should be kept in mind that this process is based on the analysis of a noiseless track, and the use of the covariances thus obtained are inherently small for the noisy-track simulations with subsequent estimator error.

<u>Run Number</u>	<u>Order of Filter</u>	<u>Special Features</u>
1	6	1 Monte-Carlo run No additive noise on track
2	6	IGAIN = 0 (gains calculated every run)
3	6	IGAIN = 1 (gains calculated on first run with noiseless track)
4	6	IGAIN = 1 Constant <u>Q</u> (values listed in Appendix B)
5	6	Same as 4 OPT = 1 (synthetic initialization) <u>P</u> (k/k-1) = 0
6	6	Same as 4 BEGIN = 1 (<u>\hat{x}</u> (0/-1) is set to 0)
7	6	IGAIN = 0 Adaptive <u>Q</u> with $K_1 = .1$, $K_2 = .9$
8	6	IGAIN = 0 Adaptive <u>Q</u> with $K_1 = .5$, $K_2 = .5$
9	6	IGAIN = 0 Adaptive <u>Q</u> with $K_1 = .9$, $K_2 = .1$
10	6	IGAIN = 0 Adaptive <u>Q</u> with $K_1 = .1$, $K_2 = .9$ IDIR = 1 (45° rotation of track)

Run Number	Order of Filter	Special Features
11	6	IGAIN = 0 Adaptive \underline{Q} with $K_1 = .5$, $K_2 = .5$ IDIR = 1
12	6	IGAIN = 0 Adaptive \underline{Q} with $K_1 = .9$, $K_2 = .1$ IDIR = 1
13	9	1 Monte-Carlo run No additive noise on track
14	9	IGAIN = 1
15	9	IGAIN = 1 Constant \underline{Q}
16	9	IGAIN = 0 Adaptive \underline{Q} with $K_1 = .1$, $K_2 = .9$
17	9	IGAIN = 0 Adaptive \underline{Q} with $K_1 = .5$, $K_2 = .5$
18	9	IGAIN = 0 Adaptive \underline{Q} with $K_1 = .9$, $K_2 = .1$
19	9	Same as 18 Only position and velocity estimates used for shell-at-target accuracy calculations
20	6-9	Model of MK-86 estimator No additive noise on track 1 Monte-Carlo run

The second column of Table II lists shell-at-target figures of merit normalized to the results obtained with the MK-86 estimator. This is done to indicate relative performance against what must be considered a severe test of the filters studied. Due to compatibility problems between the MK-86 estimator and the MCSP the results for run 20 are based on a noiseless track with one Monte-Carlo simulation. This produces optimal statistics since noise on the track degrades filter performance, and the comparison between the MK-86

estimator and the other filters in the study is slanted in favor of the former.

Additional insight into the potential success of the fire control system estimators in generating accurate firing orders is reflected in Table III where summaries of the numbers of hits within several radial miss distances of the target are tabulated for the better filters. Again, these figures would undoubtedly improve given a less maneuvering target, but it is worthwhile to know the possible consequences of a threat as reflected in the modeled track. The numbers shown are out of a total of 197 shells fired in the simulation.

B. EVALUATION OF RESULTS

An analysis of the numerical results presented in Table II reveals several items of interest. In this section the influence of such factors as the method of determining and using gains, initialization, type of \underline{Q} matrix used, and direction of approach are empirically compared, and possible explanations for the trends shown are discussed.

The motivation for runs 1-3 and 13-14 was to check the reliability of the Monte-Carlo statistics and to compare the various techniques for determining and using gains. With noise of mean zero added to the track in runs 2, 3, and 14 the filter estimate errors should have been consistent with those generated in runs 1 and 13 (single run, no additive noise simulations). In addition, runs 2 and 3 were executed

TABLE II
ESTIMATOR PERFORMANCE TABLE

Run Number	Normalized RMS Shell-at-Target Miss Distance	RMS Mean Filter Position Estima- tion Error - yds	RMS Mean Filter Velocity Estima- tion Error - yds/sec
1	5.49	463.83	79.02
2	5.48	463.67	78.99
3	5.48	463.80	78.99
4	0.93	5.19	11.22
5	0.94	5.29	11.42
6	0.93	5.22	11.22
7	2.40	8.48	58.32
8	2.06	5.88	47.92
9	0.96	2.06	16.34
10	2.40	8.91	61.18
11	2.06	6.12	50.82
12	0.96	2.12	17.46
13	3.44	190.22	50.13
14	3.46	190.32	50.26
15	1.74	1.79	6.63
16	1.62	7.13	32.74
17	1.46	4.35	26.09
18	1.19	1.59	10.25
19	0.79	1.59	10.25
20	1.00	6.30	10.43

TABLE III
NUMBER OF HITS WITHIN SEVERAL MISS
DISTANCES FOR SELECTED FILTERS

Run	Description	15	20	25	30
4	6th order Constant \underline{Q}	3	6	10	16
9	6th order Adaptive \underline{Q} $K_1 = .9, \bar{K}_2 = .1$	0	0	0	0
15	9th order Constant \underline{Q}	35	41	44	52
18	9th order Adaptive \underline{Q} $k_1 = .9, \bar{K}_2 = .1$	18	30	40	43
19	Same as 18 Accel. terms not used	0	6	13	20
20	MK-86 Estimator	0	3	6	12

to test the reliability of using gains calculated on the first run having a noiseless track against the results of a simulation having gains calculated on every run. As can be seen in Table II both tests showed desired results--the statistics for 100 runs with noisy tracks gave the expected mean errors, and the gain calculations on the first run having a noiseless track provided reliable results if used for all other runs having noisy tracks. It should be noted that calculating and using gains in this manner is only possible when \underline{Q} is a constant matrix. Attempts to use $\text{IGAIN} = 1$ with adaptive \underline{Q} simulations resulted in major discrepancies because the noise on the track is a major factor influencing the magnitude of the \underline{Q} matrix from point to point, and the noiseless track did not provide similar residuals.

Runs 4-6 were designed to evaluate the various initialization methods, and the results show only minor variation.

A most interesting trend in the application of the adaptive \underline{Q} option is revealed by runs 7-12 and 16-18. It is apparent that for the track used it is beneficial to provide some damping in the form of a high K_1/K_2 ratio. A plausible explanation is that additive noise can cause undesirable residual variation and subsequent erratic \underline{Q} matrices if allowed to too heavily weight the updated \underline{Q} matrix. Selecting the correct ratio shows obvious improvement in filter performance and provides better performance factors in Table II than those for the constant \underline{Q} runs with 6th and

9th order filters. However, the numbers of hits within the several radii shown in Table III do not show the same pattern, and this suggests that if the requirement to have a few shells within a small radius is more important than having a lot of shells within a larger radius, then a less responsive \underline{Q} matrix calculating technique is needed. Because in most airborne filtering situations the random forcing on the target is unknown, the successful implementation of an adaptive technique seems to be the only way to deal with all possible situations.

The rotation of the track in runs 10-12 was provided to check that option and confirm that the trends shown are not functions of the direction of approach of the target.

A final area of comparison is between the 6th and 9th order filters with similar options used. As expected the 9th order filter provided smaller mean filter estimation errors because of its ability to exactly follow a constantly-accelerating target. Specific runs for comparison are 4 and 15 for the constant \underline{Q} runs and 9 and 18 for the best of the adaptive \underline{Q} runs. As shown in Table II the two methods for determining \underline{Q} provide similar results for the 6th-order filter with the constant \underline{Q} method producing slightly smaller filter estimation errors. In the case of the 9th-order filter the adaptive \underline{Q} technique provides superior position estimates, but the constant \underline{Q} method is superior in generating velocity estimates.

A most noteworthy incompatibility is the contradiction shown for shell-at-target miss distances between 6th and 9th-order estimators. From Table II it is suggested that the 6th-order filters offer superior performance in minimizing the miss distances, but the actual superiority belongs to the 9th-order filters as evidenced by Table III. The discrepancy can be explained by the fact that the 6th-order filters consistently provide miss distances that are in the range from 50 to 150 yards but fail to reduce this substantially whereas the 9th-order filters do provide these minimum miss distances but are substantially worse at ranges greater than 3000 yards due to the effects of acceleration estimation errors. At great distances these acceleration estimation errors are important due to the larger flight times of the shells.

Run 19 was included as a special study to determine the shell-at-target accuracy for the filter in run 18 when only position and velocity estimates were used for prediction. Some internal changes were required in the MCSP to obtain these results that are intended to reinforce the comments made above.

The results for run 20 present the performance of the existing system against the scenario of the MCSP track. This is a combination 6th-9th order filter using only position and velocity estimates to generate predicted target location for determining gunfire orders.

VI. CONCLUSIONS AND RECOMMENDATIONS

Judging from the results presented in the previous section the techniques and derivations presented in this thesis have been shown to be applicable to airborne tracking problems with possible results superior to those obtained using present estimators. The use of a Monte-Carlo simulator has been beneficial in insuring that the results obtained are statistically legitimate for the scenario studied. Also, the capacity to vary the implementation of the many options available has made it possible to determine those factors that offer the greatest potential for improved estimator performance.

While the results presented in this thesis are encouraging, the scope of the study must be reiterated in order to avoid drawing too broad conclusions regarding the applicability to aircraft tracking problems. The track developed and used was selected because of its common use today in typical aircraft tactics, but much additional analysis would be necessary to categorically accept the relative capacities of the several filters presented herein. It is the adherence to such an analytical approach that can insure that the filters acquired for modern gunfire control systems are capable of handling the threats they are expected to experience.

The results have shown that given a complex target approach, basically simple filters are capable of surprising results if reinforced by techniques designed specifically for the model scenario. The online generation of the covariance matrix of measurement noise and adaptive covariance matrices of random forcing are examples that contribute to minimizing filter estimation errors. These features are, of course, dependent upon the availability of computation capacity, but with the advent of smaller, faster, and less expensive computers the restrictions imposed are becoming more liberal all the time.

This thesis was intended to correlate some of the important concepts pertinent to airborne tracking filters into an easily used simulation program for current and future study. Some areas that appear to offer additional potential for filter improvement and simulation accuracy include continued study in the field of adaptive covariance matrices of random forcing, a feature seen to be most powerful in this thesis, further work in the area of computation time and storage requirements for implementing various filtering approaches, and refinement of the shell-at-target accuracy calculations. Furthermore, implementation of simulation-generated techniques into actual models for testing is mandatory to provide feedback for refining the simulators, thus saving investment in inferior systems due to acceptance of incorrect simulation results.

APPENDIX A

PROGRAM LISTING FOR MONTE-CARLO RUN REQUIREMENT EVALUATION

The program written for analyzing the existing generators of normally distributed random numbers is included in this appendix. It provides statistics for empirical mean and standard deviation within 5, 10, 15, 20, and 25 percent of the desired standard deviation for ensembles taken a variable number of times.

In the program the following variables can be altered to obtain additional statistics describing expected results in Monte-Carlo simulations

ITER - The number of samples in each ensemble

N - The number of ensembles determined by data cards using an I5 format in the first five columns.

The program is written to terminate when $N = 5000$, but this can be revised by referring to the statement identified by NORM0094.


```

C      THIS PROGRAM ANALYZES THE AVAILABLE RANDCM NUMBER GENERATORS FOR
C      NORMALLY DISTRIBUTED RANDOM NUMBERS
C
C      REAL MEAN, NUM1, NUM2, NUM3, NUM4, NUM5, NUM6, NUM7, NUM8, NUM9, NUM10
C      DIMENSION MEAN(10000), STDDEV(10000)
C
C      ITER IS THE NUMBER OF TIMES THE REQUESTED NUMBER OF SAMPLES ARE
C      TAKEN TO ASSURE STEADY-STATE CONDITIONS
C
C      10 ITER = 1000
C
C      ITER IS NUMBER OF ITERATIONS OF 'N' SAMPLES EACH
C
C      25 READ(5,25) N
C      FORMAT(I5)
C
C      XMEAN IS THE REQUESTED MEAN
C      XSTD IS THE REQUESTED STANDARD DEVIATION
C
C      XMEAN = 0.0
C      XSTD = 2.0
C      WRITE(6,100) N, ITER, XMEAN, XSTD
C      100 FORMAT(I11,10X,'EXPERIMENTAL NORMAL DISTRIBUTION',/,
C      1 11X,'PARAMETERS FOR ',I3,' SAMPLES TAKEN',I5,' TIMES',/,11X,
C      2 'WITH GIVEN MEAN=',F4.0,' STD. DEV.=',F4.0,/,/)
C
C      THE FOLLOWING CARD IS NOT PRESENT WHEN 'GAUSS' IS USED
C
C      CALL OVFLOW
C      KERNEL = 2568317
C      DO 1000 I=1, ITER
C      MEAN(I) = 0
C      STDDEV(I) = 0
C      DO 2000 J=1, N
C
C      WHEN GAUSS IS USED THE CALLING STATEMENT IS 'CALL GAUSS (KERNEL,
C      XSTD, XMEAN, TEMP)', AND THE FOLLOWING TWO CARDS ARE ELIMINATED
C
C      CALL NORMAL(KERNEL, TEMP, 1)
C      TEMP = TEMP * XSTD
C      TERM1 = FLOAT(J-1)/FLOAT(J)
C      TERM2 = 1.0/FLOAT(J)
C      MEAN(I) = MEAN(I) * TERM1 + TERM2 * TEMP
C      STDDEV(I) = STDDEV(I) + (TEMP - MEAN(I)) ** 2
C      STDDEV(I) = SQRT(STDDEV(I)/(N-1))
C      NUM1=0
C      NUM2=0
C
C      2000
C      1000
C      50

```



```

NUM3=0
NUM4=0
NUM5=0
NUM6=0
NUM7=0
NUM8=0
NUM9=0
NUM10=0
DO 300 I=1,ITER
  FIND RISK(MEAN(I)-XMEAN) NUM1=NUM1+1
  DIFF=ABS(LE(.05*XSTD)) NUM2 = NUM2+1
  IF(DIFF.LE.(.15*XSTD)) NUM3=NUM3+1
  IF(DIFF.LE.(.25*XSTD)) NUM4 =NUM4+1
  IF(DIFF.LE.(.25*XSTD)) NUM5=NUM5+1
  IF(DIFF.LE.(.25*XSTD)) NUM6=NUM6+1
  IF(DIFF.LE.(.10*XSTD)) NUM7=NUM7+1
  IF(DIFF.LE.(.15*XSTD)) NUM8=NUM8+1
  IF(DIFF.LE.(.20*XSTD)) NUM9=NUM9+1
  IF(DIFF.LE.(.25*XSTD)) NUM10=NUM10+1
  CONTINUE
  FITTER=I
  R5=NUM1/FITTER*100.
  R10=NUM2/FITTER*100.
  R15=NUM3/FITTER*100.
  R25=NUM4/FITTER*100.
  S5=NUM5/FITTER*100.
  S10=NUM6/FITTER*100.
  S15=NUM7/FITTER*100.
  S25=NUM8/FITTER*100.
  S25=NUM9/FITTER*100.
  S25=NUM10/FITTER*100.
  WRITE(6,400)R5,S5,R10,S10,R15,S15,R20,S20,R25,S25
  FORMACEN INT OF SD,/,6X,PERCENT OF MEANS,11X,NT,12X,
  WITHIN PERCENT SD,/,16X,DESIRE SD,/,6X,60(,-),//,
  1 2 3 4 5 6 7 8 10
  STOP

```


APPENDIX B TRACK GENERATING PROGRAM

The following program generates the track data used in the simulations conducted for comparison in this thesis. Both position and velocity data are produced for each of the 233 points on the track, and, in addition, the means and variances of accelerations and acceleration rates are calculated and averaged over the entire track to provide data used in the "Constant \underline{Q} " simulations.

The results of the latter calculations follow, and it should be remembered that the validity of using the variances in defining a constant \underline{Q} matrix is dependent on having enough sample points to insure accelerations and acceleration rates with the assumed mean of random forcing (zero).

Mean acceleration in X direction:	-1.675	yd/sec ²
" Y "	-1.419	"
" Z "	0.772	"
Variance of acceleration in X direction:	45.739	(yd/sec ²) ²
" Y "	296.869	"
" Z "	224.467	"
Mean acceleration rate in X direction:	-0.131	yd/sec ³
" Y "	-0.351	"
" Z "	0.207	"
Variance of acceleration rate in X direction:	106.984	(yd/sec ³) ²
Variance of acceleration rate in Y direction:	957.846	"
Variance of acceleration rate in Z direction:	4249.27	"

C
C
C
C
C

THIS PROGRAM GENERATES THE TRACK AND VELOCITY DATA AS WELL AS
GENERATING THE VARIANCES USED FOR FINDING A CONSTANT,
OPTIMAL, Q MATRIX

```

IMPLICIT REAL*8 (A-H,L-Z)
DIMENSION X(233),Y(233),Z(233)
1  XVEL(233),YVEL(233),ZVEL(233)
2  ,VSTOR(233,3),ASTOR(233,3),ARSTOR(233,3)
DEGRAD = 57.2957795131D0
X(1) = 17250.D0
Y(1) = 0.D0
Z(1) = 1300.D0
XVEL(1) = -417.5D0
YVEL(1) = 0.D0
ZVEL(1) = 0.D0
THETA = 80.D0/DEGRAD
R = 6680.D0/THETA
ANGLE = 0.D0
DO 10 I=2,33
  ANGLE = ANGLE+1.25D0/DEGRAD
  ZTEMP = R*DCOS(ANGLE)
  XVEL(I) = -417.5D0*DCOS(ANGLE)
  YVEL(I) = 0.D0
  ZVEL(I) = -417.5D0*DSIN(ANGLE)
  X(I) = X(1)-XTEMP
  Y(I) = 0.D0
  Z(I) = Z(1)-(R-ZTEMP)
10

```

C
C
C

THIS IS END OF INITIAL DIVE PHASE - BEGIN ACCELERATING DIVE

```

SIN40 = DSIN(40.D0/DEGRAD)
COS40 = DCOS(40.D0/DEGRAD)
TIME = .25D0
DIST = 7500.D0/SIN40
VEL = (2.D0*DIST)/20.D0-417.5D0
DVEL = (VEL-417.5D0)/80.D0
CLDVEL = 417.5D0
DO 20 I=34,113
  X(I) = X(1-1)-(CLDVEL*TIME+DVEL*TIME/2.D0)*COS40
  Y(I) = Y(1-1)
  Z(I) = Z(1-1)-(CLDVEL*TIME+DVEL*TIME/2.D0)*SIN40
  CLDVEL = CLDVEL+DVEL
  XVEL(I) = -CLDVEL*DCOS(40.D0/DEGRAD)
  YVEL(I) = 0.D0
  ZVEL(I) = -CLDVEL*DSIN(40.D0/DEGRAD)
20

```

C

TRAC0000
TRAC0001
TRAC0002
TRAC0003
TRAC0004
TRAC0005
TRAC0006
TRAC0007
TRAC0008
TRAC0009
TRAC0010
TRAC0011
TRAC0012
TRAC0013
TRAC0014
TRAC0015
TRAC0016
TRAC0017
TRAC0018
TRAC0019
TRAC0020
TRAC0021
TRAC0022
TRAC0023
TRAC0024
TRAC0025
TRAC0026
TRAC0027
TRAC0028
TRAC0029
TRAC0030
TRAC0031
TRAC0032
TRAC0033
TRAC0034
TRAC0035
TRAC0036
TRAC0037
TRAC0038
TRAC0039
TRAC0040
TRAC0041
TRAC0042
TRAC0043
TRAC0044
TRAC0045
TRAC0046
TRAC0047

C	THIS IS END OF ACCELERATING DIVE - BEGIN PULLOUT	TRAC00048
C	R = 12.00*VEL/THETA	TRAC00049
	ANGLE = 0.00	TRAC00050
	ADEL = 40.00/24.00/DEGRAD	TRAC00051
	DO 30 I=114,137	TRAC00052
	ANGLE = ANGLE+ADEL	TRAC00053
	CPRIME = 2.00*R*DSIN(ANGLE/2.00)	TRAC00054
	PHI = 40.00/DEGRAD - ANGLE/2.00	TRAC00055
	XVEL(I) = -VEL*DCOS(40.00/DEGRAD-ANGLE)	TRAC00056
	YVEL(I) = 0.00	TRAC00057
	ZVEL(I) = -VEL*DSIN(40.00/DEGRAD-ANGLE)	TRAC00058
	X(I) = X(113)-CPRIME*DCOS(PHI)	TRAC00059
	Z(I) = Z(113)-CPRIME*DSIN(PHI)	TRAC00060
30	Y(I) = Y(I-1)	TRAC00061
		TRAC00062
C	THIS IS END OF PULLOUT - BEGIN EVASIVE MANEUVER TO RIGHT OF 5 DEG	TRAC00063
C		TRAC00064
	THETA = 5.00/DEGRAD	TRAC00065
	R = VEL / THETA	TRAC00066
	ANGLE = 0.00	TRAC00067
	DO 40 I=138,141	TRAC00068
	ANGLE = ANGLE+1.2500/DEGRAD	TRAC00069
	XVEL(I) = -VEL*DCOS(ANGLE)	TRAC00070
	YVEL(I) = VEL*DSIN(ANGLE)	TRAC00071
	ZVEL(I) = 0.00	TRAC00072
	X(I) = X(137)-R*DSIN(ANGLE)	TRAC00073
	Y(I) = Y(137)+R*DCOS(ANGLE)	TRAC00074
40	Z(I) = Z(137)	TRAC00075
		TRAC00076
C	THIS IS END OF STARBOARD MANEUVER - REPEAT TO PORT	TRAC00077
C		TRAC00078
	ANGLE = 0.00	TRAC00079
	DO 50 I=142,145	TRAC00080
	ANGLE = ANGLE+1.2500/DEGRAD	TRAC00081
	CPRIME = 2.00*R*DSIN(ANGLE/2.00)	TRAC00082
	PHI = 85.00/DEGRAD+ANGLE/2.00	TRAC00083
	XVEL(I) = -VEL*DCOS(5.00/DEGRAD-ANGLE)	TRAC00084
	YVEL(I) = VEL*DSIN(5.00/DEGRAD-ANGLE)	TRAC00085
	ZVEL(I) = 0.00	TRAC00086
	X(I) = X(141)-CPRIME*DSIN(PHI)	TRAC00087
	Y(I) = Y(141)+CPRIME*DCOS(PHI)	TRAC00088
50	Z(I) = Z(141)	TRAC00089
		TRAC00090
C	THIS IS END OF PORT MANEUVER - EXECUTE ANOTHER SIMILAR PORT MAN.	TRAC00091
C		TRAC00092
	ANGLE = 0.00	TRAC00093
	DO 60 I=146,149	TRAC00094
		TRAC00095


```

ANGLE = ANGLE+1.25*DO/DEGRAD
XVEL(I) = -VEL*DCOS(ANGLE)
YVEL(I) = -VEL*DSIN(ANGLE)
ZVEL(I) = 0.00
X(I) = X(145)-R*DSIN(ANGLE)
Y(I) = Y(145)-(R-R*DCOS(ANGLE))
Z(I) = Z(145)

```

60

CCC

THIS IS END OF PORT MANEUVER - PLANE NOW HAS PASSED OVER SHIP
EXECUTE 3.5 G PORT TURN FOR 2 SECS AND BEGIN 10 DEG CLIMB

```

SIN10 = DSIN(10.00/DEGRAD)
COS10 = DCOS(10.00/DEGRAD)
ANGLE = 0.00
GRAV = 32.2*DO
TAU = 5.00/DEGRAD
R = VEL**2/(3.5*DO*GRAV)
S = VEL**2*DO
THETA = S/R
DELTH = THETA/8.00
DC70 I=150,157
ANGLE = ANGLE+DELTH
C PHI = DO*TAU+ANGLE/2.00
XVEL(I) = -VEL*DCOS(5.00/DEGRAD+ANGLE)
YVEL(I) = -VEL*DSIN(5.00/DEGRAD+ANGLE)
ZVEL(I) = VEL*SIN10
X(I) = X(149)-C*DCOS(PHI)
Y(I) = Y(149)-C*DSIN(PHI)
Z(I) = Z(149)+C*SIN10
TAU = TAU+THETA

```

70

CCC

END OF TURN - BEGIN 3.5 G STARBOARD TURN FOR 2 SECS.

```

ANGLE = 0.00
DO 80 I=158,165
  DELTH = ANGLE+DELTH
  C PHI = DO*TAU+ANGLE/2.00
  XVEL(I) = -VEL*DCOS(5.00/DEGRAD+ANGLE)
  YVEL(I) = -VEL*DSIN(5.00/DEGRAD+ANGLE)
  ZVEL(I) = VEL*SIN10
  X(I) = X(157)-C*DCOS(PHI)
  Y(I) = Y(157)-C*DSIN(PHI)
  Z(I) = Z(157)+C*SIN10
  TAU = TAU+THETA

```

80

CCC

END OF TURN - BEGIN 3.0 G PORT TURN FOR 2 SECS.

TRAC00096
TRAC00097
TRAC00098
TRAC00099
TRAC00100
TRAC00101
TRAC00102
TRAC00103
TRAC00104
TRAC00105
TRAC00106
TRAC00107
TRAC00108
TRAC00109
TRAC00110
TRAC00111
TRAC00112
TRAC00113
TRAC00114
TRAC00115
TRAC00116
TRAC00117
TRAC00118
TRAC00119
TRAC00120
TRAC00121
TRAC00122
TRAC00123
TRAC00124
TRAC00125
TRAC00126
TRAC00127
TRAC00128
TRAC00129
TRAC00130
TRAC00131
TRAC00132
TRAC00133
TRAC00134
TRAC00135
TRAC00136
TRAC00137
TRAC00138
TRAC00139
TRAC00140
TRAC00141
TRAC00142
TRAC00143

C

```

ANGLE = Q.DO
R = VEL**2/(3.DO*GRAV)
THETA = S/R
DELTH I = 166,173
DO 90 I = 166,173
  ANGLE = ANGLE+DELTH
  C = 2.DO*R*DSIN(ANGLE/2.DO)
  PHI = TAU+ANGLE/2.DO
  XVEL(I) = -VEL*DCOS(TAU+ANGLE)
  YVEL(I) = -VEL*DSIN(TAU+ANGLE)
  ZVEL(I) = VEL*SIN10
  X(I) = X(165)-C*DCOS(PHI)
  Y(I) = Y(165)-C*DSIN(PHI)
  Z(I) = Z(165)+C*SIN10
  TAU = TAU+THETA

```

90

END OF TURN -- BEGIN 3.0 G STARBOARD TURN FOR 2 SECS.

C C

```

ANGLE = 0.DO
DO 100 I = 174,181
  ANGLE = ANGLE+DELTH
  C = 2.DO*R*DSIN(ANGLE/2.DO)
  PHI = 90.DO/DEGRAD-TAU+ANGLE/2.DO
  XVEL(I) = -VEL*DCOS(TAU+ANGLE)
  YVEL(I) = -VEL*DSIN(TAU+ANGLE)
  ZVEL(I) = VEL*SIN10
  X(I) = X(173)-C*DSIN(PHI)
  Y(I) = Y(173)-C*DCOS(PHI)
  Z(I) = Z(173)+C*SIN10
  TAU = TAU-THETA

```

100

END OF TURN -- BEGIN 3.0 G PORT TURN FOR 3 SECS.

C C C

```

ANGLE = 0.DO
S = VEL**3.DO
THETA = S/R
DELTH I = 182,193
DO 110 I = 182,193
  ANGLE = ANGLE+DELTH
  C = 2.DO*R*DSIN(ANGLE/2.DO)
  PHI = TAU+ANGLE/2.DO
  XVEL(I) = -VEL*DCOS(TAU+ANGLE)
  YVEL(I) = -VEL*DSIN(TAU+ANGLE)
  ZVEL(I) = VEL*SIN10
  X(I) = X(181)-C*DCOS(PHI)
  Y(I) = Y(181)-C*DSIN(PHI)
  Z(I) = Z(181)+C*SIN10

```

110

TRAC0144
TRAC0145
TRAC0146
TRAC0147
TRAC0148
TRAC0149
TRAC0150
TRAC0151
TRAC0152
TRAC0153
TRAC0154
TRAC0155
TRAC0156
TRAC0157
TRAC0158
TRAC0159
TRAC0160
TRAC0161
TRAC0162
TRAC0163
TRAC0164
TRAC0165
TRAC0166
TRAC0167
TRAC0168
TRAC0169
TRAC0170
TRAC0171
TRAC0172
TRAC0173
TRAC0174
TRAC0175
TRAC0176
TRAC0177
TRAC0178
TRAC0179
TRAC0180
TRAC0181
TRAC0182
TRAC0183
TRAC0184
TRAC0185
TRAC0186
TRAC0187
TRAC0188
TRAC0189
TRAC0190
TRAC0191


```

C
C
C
TAU = TAU+THETA
END OF TURN - BEGIN 2.5 G STARBOARD TURN FOR 3 SECS.
ANGLE = 0.00
R = VEL**2/(2.50*GRAV)
THETA = S/R
DELTH = THETA/12.00
DO 120 I=194,205
  ANGLE = ANGLE+DELTH
  C = 2.00*R*DSIN(ANGLE/2.00)
  PHI = 90.00/DEGRAD - TAU+ANGLE/2.00
  XVEL(I) = -VEL*COS10*DCOS(TAU-ANGLE)
  YVEL(I) = -VEL*COS10*DSIN(TAU-ANGLE)
  ZVEL(I) = VEL*SIN10
  X(I) = X(193)-C*DCOS(PHI)
  Y(I) = Y(193)-C*DCOS(PHI)
  Z(I) = Z(193)+C*SIN10
120 TAU = TAU-THETA
END OF TURN - BEGIN 2.5 G PORT TURN FOR 4 SECS.
ANGLE = 0.00
S = VEL*4.00
THETA = S/R
DELTH = THETA/16.00
DO 130 I=206,221
  ANGLE = ANGLE+DELTH
  C = 2.00*R*DSIN(ANGLE/2.00)
  PHI = TAU+ANGLE/2.00
  XVEL(I) = -VEL*COS10*DCOS(TAU+ANGLE)
  YVEL(I) = -VEL*COS10*DSIN(TAU+ANGLE)
  ZVEL(I) = VEL*SIN10
  X(I) = X(205)-C*DCOS(PHI)
  Y(I) = Y(205)-C*DCOS(PHI)
  Z(I) = Z(205)+C*SIN10
130 TAU = TAU+THETA
END OF TURN - BEGIN 2.0 G STARBOARD TURN FOR 3 SECS.
ANGLE = 0.00
S = VEL*3.00
R = VEL**2/(2.00*GRAV)
THETA = S/R
DELTH = THETA/12.00
DO 140 I=222,233
  ANGLE = ANGLE+DELTH
  C = 2.00*R*DSIN(ANGLE/2.00)

```

TRAC0192
 TRAC0193
 TRAC0194
 TRAC0195
 TRAC0196
 TRAC0197
 TRAC0198
 TRAC0199
 TRAC0200
 TRAC0201
 TRAC0202
 TRAC0203
 TRAC0204
 TRAC0205
 TRAC0206
 TRAC0207
 TRAC0208
 TRAC0209
 TRAC0210
 TRAC0211
 TRAC0212
 TRAC0213
 TRAC0214
 TRAC0215
 TRAC0216
 TRAC0217
 TRAC0218
 TRAC0219
 TRAC0220
 TRAC0221
 TRAC0222
 TRAC0223
 TRAC0224
 TRAC0225
 TRAC0226
 TRAC0227
 TRAC0228
 TRAC0229
 TRAC0230
 TRAC0231
 TRAC0232
 TRAC0233
 TRAC0234
 TRAC0235
 TRAC0236
 TRAC0237
 TRAC0238
 TRAC0239

TRAC0240
TRAC0241
TRAC0242
TRAC0243
TRAC0244
TRAC0245
TRAC0246
TRAC0247
TRAC0248
TRAC0249
TRAC0250
TRAC0251
TRAC0252
TRAC0253
TRAC0254
TRAC0255
TRAC0256
TRAC0257
TRAC0258
TRAC0259
TRAC0260
TRAC0261
TRAC0262
TRAC0263
TRAC0264
TRAC0265
TRAC0266
TRAC0267
TRAC0268
TRAC0269
TRAC0270
TRAC0271
TRAC0272
TRAC0273
TRAC0274
TRAC0275
TRAC0276
TRAC0277
TRAC0278
TRAC0279
TRAC0280
TRAC0281
TRAC0282
TRAC0283
TRAC0284
TRAC0285
TRAC0286
TRAC0287

```

PHI = 90.00/DEGRAD-TAU+ANGLE/2.00
XVEL(I) = -VEL*DCOS10*DCOS(TAU-ANGLE)
YVEL(I) = -VEL*DCOS10*DSIN(TAU-ANGLE)
ZVEL(I) = VEL*SIN10
X(I) = X(221)-C*DCOS(PHI)
Y(I) = Y(221)-C*DCOS(PHI)
Z(I) = Z(221)+C*SIN10
DC 150 I=1,233
140

```

C
C
C
C

CHANGE POSITION AND VELOCITY DATA TO YARDS AND YARDS/SEC
INSTEAD OF FEET AND FEET/SEC

```

X(I) = X(I)/3.00
Y(I) = Y(I)/3.00
Z(I) = Z(I)/3.00
VEL = DSQRT(XVEL(I)**2+YVEL(I)**2+ZVEL(I)**2)
XVEL(I) = XVEL(I)/3.00
YVEL(I) = YVEL(I)/3.00
ZVEL(I) = ZVEL(I)/3.00

```

C
C
C
C
C

THE POSITION/VELOCITY VALUES ARE PUNCHED ON CARD PAIRS WITH
THE 3 POSITION VALUES ON THE FIRST CARD AND THE 3 VELOCITY
VALUES ON THE SECOND

```

150 WRITE(7,4778) I,X(I),Y(I),Z(I),I,XVEL(I),YVEL(I),ZVEL(I)
4778 FORMAT(5X,I3,3D20.10,/,5X,I3,3D20.10)
4777 FCORMAT(5X,I3,3D20.10,/,5X,I3,3D20.10,10X,D20.6)

```

C
C
C
C
C

THE REMAINDER OF THE PROGRAM GENERATES VARIANCES OF ACCELERATION
AND ACCELERATION RATES BY TAKING SUCCESSIVE DIFFERENCES AND
ANALYZING OVER THE ENTIRE TRACK

```

160 DO 160 I=1,232
VSTOR(I,1) = (X(I+1)-X(I))/ .2500
VSTOR(I,2) = (Y(I+1)-Y(I))/ .2500
VSTOR(I,3) = (Z(I+1)-Z(I))/ .2500
DC 170 I=1,231
ASTOR(I,1) = (VSTOR(I+1,1)-VSTOR(I,1))/ .2500
ASTOR(I,2) = (VSTOR(I+1,2)-VSTOR(I,2))/ .2500
ASTOR(I,3) = (VSTOR(I+1,3)-VSTOR(I,3))/ .2500
170 DO 180 I=1,230
ARSTOR(I,1) = (ASTOR(I+1,1)-ASTOR(I,1))/ .2500
ARSTOR(I,2) = (ASTOR(I+1,2)-ASTOR(I,2))/ .2500
ARSTOR(I,3) = (ASTOR(I+1,3)-ASTOR(I,3))/ .2500
180 ARMEANX=0.00
AMEANY = 0.00
AMEANZ = 0.00

```



```

AVARX = 0.00
AVARY = 0.00
AVARZ = 0.00
DC 190 I=1,231
AMEANX = AMEANX+ASTOR(I,1)
AMEANY = AMEANX+ASTOR(I,2)
AMEANZ = AMEANX+ASTOR(I,3)
AVARX = AVARX+ASTOR(I,1)**2
AVARY = AVARX+ASTOR(I,2)**2
AVARZ = AVARX+ASTOR(I,3)**2
190 AMEANX = AMEANX/231.00
AMEANY = AMEANX/231.00
AMEANZ = AMEANX/231.00
AVARX = AVARX/231.00-AMEANX**2
AVARY = AVARX/231.00-AMEANY**2
AVARZ = AVARX/231.00-AMEANZ**2
WRITE(6,2719) AMEANX,AMEANY,AMEANZ,AVARX,AVARY,AVARZ
2719 FORMAT(1H1,9X,'ACC MEAN X=',D20.6,'/',10X,'VAR X=',D20.6,'/',10X,'ACC MEAN Y=',D20.6,'/',
1 10X,'ACC MEAN Z=',D20.6,'/',10X,'VAR Z=',D20.6,'/',10X,'ACC RATE MEAN X=',D20.6,'/',10X,
3 ;/,10X,'VAR X=',D20.6,'/',10X,'ACC RATE MEAN Y=',D20.6,'/',10X,'VAR Y=',D20.6,'/',10X,
AMEANX = 0.00
AMEANY = 0.00
AMEANZ = 0.00
AVARX = 0.00
AVARY = 0.00
AVARZ = 0.00
DC 200 I=1,230
AMEANX = AMEANX+ARSTOR(I,1)
AMEANY = AMEANX+ARSTOR(I,2)
AMEANZ = AMEANX+ARSTOR(I,3)
AVARX = AVARX+ARSTOR(I,1)**2
AVARY = AVARX+ARSTOR(I,2)**2
AVARZ = AVARX+ARSTOR(I,3)**2
200 AMEANX = AMEANX/230.00
AMEANY = AMEANX/230.00
AMEANZ = AMEANX/230.00
AVARX = AVARX/230.00-AMEANX**2
AVARY = AVARX/230.00-AMEANY**2
AVARZ = AVARX/230.00-AMEANZ**2
WRITE(6,2720) AMEANX,AMEANY,AMEANZ,AVARX,AVARY,AVARZ
2720 FORMAT(1H0,9X,'ACC MEAN X=',D20.6,'/',10X,'VAR X=',D20.6,'/',10X,'ACC RATE MEAN Y=',
1 D20.6,'/',10X,'VAR Y=',D20.6,'/',10X,'ACC RATE MEAN Z=',D20.6,'/',10X,'VAR Z=',D20.6,'/',
2 SICP
END
TRAC0288
TRAC0289
TRAC0290
TRAC0291
TRAC0292
TRAC0293
TRAC0294
TRAC0295
TRAC0296
TRAC0297
TRAC0298
TRAC0299
TRAC0300
TRAC0301
TRAC0302
TRAC0303
TRAC0304
TRAC0305
TRAC0306
TRAC0307
TRAC0308
TRAC0309
TRAC0310
TRAC0311
TRAC0312
TRAC0313
TRAC0314
TRAC0315
TRAC0316
TRAC0317
TRAC0318
TRAC0319
TRAC0320
TRAC0321
TRAC0322
TRAC0323
TRAC0324
TRAC0325
TRAC0326
TRAC0327
TRAC0328
TRAC0329
TRAC0330
TRAC0331
TRAC0332

```


APPENDIX C

LISTING OF MONTE-CARLO SIMULATION PROGRAM (MSCP)

A complete listing of the Monte-Carlo Simulation Program is included herein. For user convenience the statement numbers are in ascending order allowing for easy location of designated statements, and the cards are labeled in the right margin for reference and ordering if required.

Variable and array descriptions appear at the beginning of the program with instructions for the input parameters immediately following each READ statement. Format statements controlling headings on the output plots are included in the program statements MCSP0600-MSCP0694.

CCCCCCCC

THIS IS THE MONTE-CARLO SIMULATION PROGRAM (MCSP)
AND CAN ANALYZE UP TO A 233 POINT TRACK AS MANY AS
99999 TIMES TO OBTAIN TRUE MCNTE-CARLO STATISTICS
OF FILTER PERFORMANCE

```
IMPLICIT REAL*8(Q,H,R,G,P,A,B,C,D,S,I)
1 REAL*8 I,X,Y,Z,XTRUE,YTRUE,ZTRUE,EST,RTRUE,BTRUE,ETRT,BT,ET,X
1 REAL*4 SEABS,SEED
INTEGRATION PRE(9,9), LABEL(10), STATES(9,9), RADAR(9,9), XTRUE(233
1), YTRUE(233), ZTRUE(233), QP(9,9), PKKMP(9,9), A(9,9),
2B(9,9), GSTR(9,9), TVELX(233), TVELY(233), TVELZ(233), RTRUE(
3233), BTRUE(233), ETRUE(233), SEABS(233), LOCK(14),
+START(9), GAMMA(9,9), GAMMAT(9,9), HCLD(9,9),
CCMCQN I(9,9), Q(9,9), R(9,9), G(9,9), PHI(9,9), PKK(9,9), PKKM1
1(9,9), RMEAN(233,6), RSID(6), RESID(233,4), AHEAD(233), EST(9,233), PHIPR
2M(9,9,40), PKSTOR(9,233)
```

DESCRIPTION OF ARRAYS FOLLOWS:

```
PRE(9,9) - FILTERS DESCRIPTION OF RUN
LABELS - USER'S DESCRIPTION OF RUN
STATES - FILTERS DESCRIPTION OF RUN
RADAR - MEASUREMENTS INPUT 'X' VALUES
XTRUE - NOISELESS INPUT 'Y' VALUES
YTRUE - NOISELESS INPUT 'Z' VALUES
ZTRUE - NOISELESS INPUT 'X' VALUES
QP - COVARIANCE MATRIX COVARIANCE MATRIX EXCITATION
PKK - COVARIANCE MATRIX COVARIANCE MATRIX EXCITATION
A - COVARIANCE MATRIX COVARIANCE MATRIX EXCITATION
B - COVARIANCE MATRIX COVARIANCE MATRIX EXCITATION
GSTR - COVARIANCE MATRIX COVARIANCE MATRIX EXCITATION
TVELX - COVARIANCE MATRIX COVARIANCE MATRIX EXCITATION
TVELY - COVARIANCE MATRIX COVARIANCE MATRIX EXCITATION
TVELZ - COVARIANCE MATRIX COVARIANCE MATRIX EXCITATION
RTRUE - COVARIANCE MATRIX COVARIANCE MATRIX EXCITATION
BT - COVARIANCE MATRIX COVARIANCE MATRIX EXCITATION
ETRT - COVARIANCE MATRIX COVARIANCE MATRIX EXCITATION
```

```
CALL QVFLOW
KERNL1=3790437
KERNL2=30117221
KERNL3=1993183
DEGRAD=57.295779513100
```

II - IDENTITY MATRIX

CC


```

CCCCCCCCCCCCCCCCCCCC
1  COVARIANCE MATRIX OF STATE EXCITATION (VARIABLE)
2  GAMMA - MATRIX TO RELATE NON-DETERMINISTIC
3  FORCING TO STATES
4  H - COVARIANCE OF MEASUREMENT ERROR MATRIX (VARIABLE)
5  G - OPTIMUM GAIN IN MATRIX (VARIABLE)
6  PHI - ESTIMATION COVARIANCE DYNAMICS P (K/K) (VARIABLE)
7  PKKM1 - ESTIMATION COVARIANCE LATER MEAN OF ERROR
8  RMEAN - MEAN OF POSITION RESIDUALS IN EACH DIRECTION PLUS TOTAL MISS
9  RSD - SUM OF POSITION RESIDUALS TO GET ESTIMATES
10 RESID - TIMES FOR SHELL FILTER POWERS OF PHI MATRIX UP TO 40
11 AHEAD - STORAGE FOR FILTER VARIANCES
12 PHIPRM - STORAGE FOR POWERS OF PHI MATRIX UP TO 40
13 PKSTOR - STORAGE FOR FILTER VARIANCES
14 I=1,233
15 DO 1 K=1,9
16 PKSTOR(K,I)=0.00
17 DO 2 J=1,6
18 RMEAN(I,J)=0.00
19 DO 3 L=1,4
20 RESID(I,L)=0.00
21 DO 4 I=1,6
22 RSD(I)=0.00
23 READ (5,6) LOOP,KVEL,(LOOK(I),I=1,14),TIME
24 FCFORMAT (2I10,10X,14I1,6X,D20.0)
25
26 LOOP IS THE NUMBER OF MONTE CARLO RUNS
27 KVEL = 1 WILL SET UP THE PROGRAM FOR READING IN TRUE VELOCITIES
28 IN ADDITION TO TRACK POSITIONS TO BE PRINTED
29 ARRAY LOOK, DETERMINES PLOTS TO BE PRINTED
30 IF INPUT ARRAY ELEMENT IS NONZERO, THE FOLLOWING PLOTS
31 WILL BE PRINTED
32 LOOK(1): MEAN DISTANCE WHEN SHELL ARRIVES
33 LOOK(2): MEAN ERROR IN X POSITION
34 LOOK(3): MEAN ERROR IN Y POSITION
35 LOOK(4): MEAN ERROR IN Z POSITION
36 LOOK(5): X POSITION VARIANCE
37 LOOK(6): Y POSITION VARIANCE
38 LOOK(7): Z POSITION VARIANCE
39 LOOK(8): MEAN VELOCITY ERROR IN X DIRECTION
40 LOOK(9): MEAN VELOCITY ERROR IN Y DIRECTION
41 LOOK(10): MEAN VELOCITY ERROR IN Z DIRECTION

```



```

CCCCCCCC
IDIR = 3      ANGLE = 202.5 DEGS
IGUN=1 WILL CALCULATE SHELL AT TARGET RESIDUALS
TRAN SX = TRANSLATION OF ALL X DATA POINTS
TRAN SY = TRANSLATION OF ALL Y DATA POINTS
TRAN SZ = TRANSLATION OF ALL Z DATA POINTS

M=N/3
IF (OPT.EQ.1.AND.N.EQ.9) OPTNO=3
IF (OPT.EQ.1.AND.N.EQ.6) OPTNO=2
NUN=N/M
N2=N/3+1
N3=N*2/3+1
N4=N2+1
N5=N3+1
DO 10 I=1,9
  DO 10 J=1,9
    RADCAR(I,J)=0.00
  STATES(I,J)=0.00
  PRED(I,J)=0.00
  GP(I,J)=0.00
  PHI(I,J)=0.00
  PKK(I,J)=0.00
  PKKMLP(I,J)=0.00
  G(I,J)=0.00
  H(I,J)=0.00
  HCLD(I,J)=0.00
  GAMMA(I,J)=0.00
  GAMMAT(I,J)=0.00
  IF (IGAIN.EQ.0.OR.IGAIN.EQ.1) GO TO 12
  DO 11 I=1,NN
    IF GAINS ARE TO BE READ, THEY ARE DONE SO HERE - THERE WILL BE
    NN QUANTITY OF (N X NUM) MATRICES
    CALL MREAD1 (G,N,NUM)
    DO 11 J=1,N
      DO 11 K=1,NUM
        GSTOR(J,K,I)=G(J,K)
      DO 13
    11
  INITIAL CONDITIONS ON COVARIANCE OF ESTIMATION ERROR READ HERE
  THIS WILL BE A (N X N) MATRIX
  12 IF(OPT.NE.1) CALL MREAD(PKKMLP,N,N)
    IF THE 'R' MATRIX IS NOT TIME VARYING, READ IT IN HERE
    13

```



```

CC      THIS WILL BE A (NUM X NUM) MATRIX
CC      IF(IR.NE.0) CALL MREAD(RP,NUM,NUM)
CC      READ INITIAL CONDITION VARIANCES
CC      OF RANDOM FORCING MATRIX
CC      THIS WILL BE A (NUM X NUM) MATRIX
CC      CALL MREAD (QP,NUM,NUM)
CC      READ GAMMA MATRIX - THIS WILL BE A (N X NUM) MATRIX
CC      CALL MREAD(GAMMA,N,NUM)
CC      READ PHI MATRIX HERE - THIS WILL BE A (N X N) MATRIX
CC      13 CALL MREAD (PHI,N,N)
CC      READ MEASUREMENT MATRIX HERE - THIS WILL BE A (NUM X N) MATRIX
CC      CALL MREAD (H,NUM,N)
CC      IF (IDIR.EQ.0) DEL=0.D0
CC      IF (IDIR.EQ.1) DEL=45.D0/DEGRAD
CC      IF (IDIR.EQ.2) DEL=90.D0/DEGRAD
CC      IF (IDIR.EQ.3) DEL=202.5D0/DEGRAD
CC      SINDEL=DSIN(DEL)
CC      CCSDEL=DCOS(DEL)
CC      DC 15 I=1,NN
CC      THE FOLLOWING SECTION READS THE TRUE (X,Y,Z) DATA AND
CC      CALCULATES TRUE (R,B,E)
CC      14 READ (5,14) XTRUE(I),YTRUE(I),ZTRUE(I)
CC      FORMAT (8X,3D20.0)
CC      XTRUE(I)=XTRUE(I)+TRANSX
CC      YTRUE(I)=YTRUE(I)+TRANSY
CC      ZTRUE(I)=ZTRUE(I)+TRANSZ
CC      XT=XTRUE(I)
CC      YT=YTRUE(I)
CC      ZT=ZTRUE(I)
CC      RT=CSQRT(XT**2+YT**2+ZT**2)
CC      BT=DATAN2(ZT,DSQRT(XT**2+YT**2))
CC      IF (YT.GE.0.D0.AND.XT.NE.0.D0) BT=DATAN2(YT,XT)+DEL
CC      IF (YT.GT.0.D0.AND.XT.EQ.0.D0) BT = PI/2.D0 + DEL
CC      IF (YT.EQ.0.D0.AND.XT.LT.0.D0) BT = 180.D0/DEGRAD + DEL
CC      IF (YT.LT.0.D0.AND.XT.LT.0.D0) BT=DATAN2(DABS(YT),DABS(XT))+DEL+180.D0/DEGRAD
CC      10.D0/DEGRAD

```



```

C      THE FOLLOWING LOOP IS THE MONTE CARLO SIMULATION
C
C 58 ITER=1,LOOP
C 47 I=1,N
C 47 J=1,N
C      G(I,J)=GP(I,J)
C      PKKM1(I,J)=PKKM1P(I,J)
C      R(I,J)=RP(I,J)
C 47 R(I,J)=RP(I,J)
C 48 I=1,N
C 48 PRED(I,1)=START(I)
C
C      THE FOLLOWING LOOP STEPS THROUGH THE TRACK POINTS
C
C 57 L=1,NN
C
C      IF DESIRED, NOISE IS ADDED TO THE TRACK POINTS
C
C      CALL NOISE (RTRUE(L),BTRUE(L),ETRUE(L),RR,RE,RE,KERNL1,
C      1 KERNL2,KERNL3,X,Y,Z,INDEX)
C      IFLAG = 0
C
C      THE FOLLOWING CARD SETS THE OPTION FOR USING INITIAL OBSERVATIONS
C      TO GENERATE PREDICTIONS INSTEAD OF FILTER ESTIMATES
C
C      IF (OPT.EQ.1.AND.L.LE.OPTNO) IFLAG=1
C
C      THE FOLLOWING SECTION HANDLES THE VARIOUS GAIN OPTIONS
C
C      IF (IGAIN.EQ.0) GO TO 52
C      IF (IGAIN.EQ.1.AND.ITER.NE.1) GO TO 50
C      IF (IGAIN.EQ.2) GO TO 50
C      CALL GAIN (N,NUM,ITER,L,IFLAG,KVEL)
C 49 IZ=1,N
C 49 IX=1,NUM
C      GSTOR(IZ,IX,L)=G(IZ,IX)
C 50 GO TO 53
C 51 IZ=1,N
C 51 IX=1,NUM
C      G(IZ,IX)=GSTOR(IZ,IX,L)
C 52 GO TO 53
C      CALL GAIN (N,NUM,ITER,L,IFLAG,KVEL)
C
C      FOR MOST SITUATIONS THE RADAR MEASUREMENTS ARE MADE NOISY
C
C 53 RADAR(1,1)=X
C      RADAR(2,1)=Y
C      RADAR(3,1)=Z
C

```

MCSP0382
 MCSP0383
 MCSP0384
 MCSP0385
 MCSP0386
 MCSP0387
 MCSP0388
 MCSP0389
 MCSP0390
 MCSP0391
 MCSP0392
 MCSP0393
 MCSP0394
 MCSP0395
 MCSP0396
 MCSP0397
 MCSP0398
 MCSP0399
 MCSP0400
 MCSP0401
 MCSP0402
 MCSP0403
 MCSP0404
 MCSP0405
 MCSP0406
 MCSP0407
 MCSP0408
 MCSP0409
 MCSP0410
 MCSP0411
 MCSP0412
 MCSP0413
 MCSP0414
 MCSP0415
 MCSP0416
 MCSP0417
 MCSP0418
 MCSP0419
 MCSP0420
 MCSP0421
 MCSP0422
 MCSP0423
 MCSP0424
 MCSP0425
 MCSP0426
 MCSP0427
 MCSP0428
 MCSP0429


```

3  7X,'DISTANCE',/,55X,'TRACK TIME',/,55X,'WHEN FIRED',/,1X,
4  88(' '),//)
    THE FOLLOWING SECTION DOES FINAL COMPUTATIONS ON STATISTICAL
    QUANTITIES AND OUTPUTS THEM
60 DC 62 I=1,NN
    RMEAN'S ARE CHANGED TO MEAN OF ESTIMATE AT EACH POINT FROM SUM
    OF ESTIMATES FOR ALL MONTE CARLO RUNS
    DC 61 J=1,5
61  RMEAN(I,J)=RMEAN(I,J)/FLOAT(LLOOP)
    THE FOLLOWING 6 TERMS ARE USED TO CALCULATE FILTER PERFORMANCE
    FACTORS, AND REPRESENT SUMS OF MEAN ERRORS
    RSD(1)=RSD(1)+DABS(RMEAN(I,1)-XTRUE(I))
    RSD(2)=RSD(2)+DABS(RMEAN(I,2)-YTRUE(I))
    RSD(3)=RSD(3)+DABS(RMEAN(I,3)-ZTRUE(I))
    RSD(4) = RSD(4) + DABS(RMEAN(I,4)-TVELX(I))
    RSD(5) = RSD(5) + DABS(RMEAN(I,5)-TVELY(I))
    RSD(6) = RSD(6) + DABS(RMEAN(I,6)-TVELZ(I))
62  CONTINUE
    IF (IGUN.NE.1) GO TO 66
    IFIRE = 0
    DC 63 I=1,NN
    RESID(I,1)=RESID(I,1)/FLOAT(LLOOP)
    RESID(I,2)=RESID(I,2)/FLOAT(LLOOP)
    RESID(I,3)=RESID(I,3)/FLOAT(LLOOP)
    RESID(I,4)=DSQRT(RESID(I,1)**2+RESID(I,2)**2+RESID(I,3)**2)
    IF INSUFFICIENT TIME HAD PASSED TO CALCULATE MISS DISTANCE
    THEN DO NOT OUTPUT STATISTICS
    IF (RESID(I,4).EQ.0.00) GO TO 63
    IFIRE = IFIRE + 1
    CALL STUDY (I,XTRUE(I),YTRUE(I),ZTRUE(I),RESID(I,1),RESID(I,2),RES
1  ID(I,3),AHEAD(I),RESID(I,4))
63  CONTINUE = 0.00
    DC 64 I=1,NN
    GUNSUM = GUNSUM + RESID(I,4)
    GMISS = GUNSUM/FLOAT(IFIRE)
64  WRITE (6,65) GMISS
65  FORMAT(1H0,///,10X,'AVERAGE SHELL AT TARGET MISS DISTANCE =',
1  D20.8,' YARDS.')
66  PERFX=RSD(1)/FLOAT(NN)

```

MCSP04
 MCSP0479
 MCSP0480
 MCSP0481
 MCSP0482
 MCSP0483
 MCSP0484
 MCSP0485
 MCSP0486
 MCSP0487
 MCSP0488
 MCSP0489
 MCSP0490
 MCSP0491
 MCSP0492
 MCSP0493
 MCSP0494
 MCSP0495
 MCSP0496
 MCSP0497
 MCSP0498
 MCSP0499
 MCSP0500
 MCSP0501
 MCSP0502
 MCSP0503
 MCSP0504
 MCSP0505
 MCSP0506
 MCSP0507
 MCSP0508
 MCSP0509
 MCSP0510
 MCSP0511
 MCSP0512
 MCSP0513
 MCSP0514
 MCSP0515
 MCSP0516
 MCSP0517
 MCSP0518
 MCSP0519
 MCSP0520
 MCSP0521
 MCSP0522
 MCSP0523
 MCSP0524
 MCSP0525


```

PERFY=RSD(2)/FLOAT(NN)
PERFZ=RSD(3)/FLOAT(NN)
PERFXV = RSD(4)/FLOAT(NN)
PERFYV = RSD(5)/FLOAT(NN)
PERFZV = RSD(6)/FLOAT(NN)

    CALCULATE FILTER PERFORMANCE FACTOR

PERF=DSQRT(PERFX**2+PERFY**2+PERFZ**2)
WRITE (6,67) PERF
67 FORMAT (1H0, '//', 10X, 'PERFORMANCE FACTOR CF FILTER, AVERAGE FILTER
1 POSITION ESTIMATE ERROR = ', D20.8, ' YARDS',)
1 PERFYV = DSQRT(PERFXV**2+PERFYV**2+PERFZV**2)
IF(KVEL.EQ.1) WRITE(6,120) PERFV
WRITE (6,68)
68 FORMAT (1H1, 'POINT NG', 6X, 'POSITION', 12X, 'VELOCITY',
1, 11X, 'VELOCITY VARIANCES',
2, /, 15X, 'ESTIMATE', 12X, '/ERRORS', 11X,
3, 'AROUND ESTIMATE', /, 16X, 'AROUND TRUTH', /,
4 88(' ',), //)
DO 73 I=1,NN
DO 69 J=1,9
69 PKSTOR(J,I)=PKSTOR(J,I)/FLOAT(LCOP)

    RMEAN'S NOW BECOME THE EXPECTED VALUE OF ESTIMATION ERROR

RMEAN(I,1)=RMEAN(I,1)-XTRUE(I)
RMEAN(I,2)=RMEAN(I,2)-YTRUE(I)
RMEAN(I,3)=RMEAN(I,3)-ZTRUE(I)

    PKSTOR'S BECOME VARIANCES DEFINED AS EXPECTED VALUE OF SQUARE
    OF ESTIMATION ERROR MINUS EXPECTED VALUE OF ESTIMATION ERROR**2
    E(X**2) - E(X)**2

PKSTOR(1,I)=PKSTOR(1,I)-RMEAN(I,1)**2
PKSTOR(N2,I)=PKSTOR(N2,I)-RMEAN(I,2)**2
PKSTOR(N3,I)=PKSTOR(N3,I)-RMEAN(I,3)**2
IF (KVEL.NE.1) GO TO 71
RMEAN(I,4)=RMEAN(I,4)-TVELX(I)
RMEAN(I,5)=RMEAN(I,5)-TVELZ(I)
RMEAN(I,6)=RMEAN(I,6)-TVELY(I)
PKSTOR(2,I)=PKSTOR(2,I)-RMEAN(I,4)**2
PKSTOR(3,I)=PKSTOR(3,I)-RMEAN(I,5)**2
PKSTOR(N4,I)=PKSTOR(N4,I)-RMEAN(I,6)**2
PKSTOR(N5,I)=PKSTOR(N5,I)-RMEAN(I,7)**2
WRITE (6,70) 1, RMEAN(I,1), RMEAN(I,4), PKSTOR(2,I), RMEAN(I,5),
1(I,2), PKSTOR(N2,I), RMEAN(I,6), PKSTOR(N3,I), RMEAN(I,7),
2, RMEAN(I,3), PKSTOR(N3,I), RMEAN(I,4), PKSTOR(N4,I), RMEAN(I,5),
70 FORMAT (1H0, 15, 4X, 'X:', D15.6, 4X, 'X:', D15.6, 4X, 'X:', D15.6, /,

```

CC

CC

CCCC


```

84 FORMAT (1H1,10X,'KETRON MK-86 THESIS',/,11X,
1 'DC SEED D(I)=RMEAN(I,3)
85 SEED D(I)=RMEAN(I,3)
86 CALL PLCTP (SEEABS,SEED,NN,0)
87 IF (LOOK(5).EQ.0) GO TO 89
88 WRITE (6,87)
89 FORMAT (1H1,10X,'KETRON MK-86 THESIS',/,11X,
1 'DC SEED D(I)=PKSTOR(1,I)
90 SEED D(I)=PKSTOR(1,I)
91 CALL PLCTP (SEEABS,SEED,NN,0)
92 IF (LOOK(6).EQ.0) GO TO 92
93 WRITE (6,90)
94 FORMAT (1H1,10X,'KETRON MK-86 THESIS',/,11X,
1 'DC SEED D(I)=PKSTOR(2,I)
95 SEED D(I)=PKSTOR(2,I)
96 CALL PLCTP (SEEABS,SEED,NN,0)
97 IF (LOOK(7).EQ.0) GO TO 95
98 WRITE (6,93)
99 FORMAT (1H1,10X,'KETRON MK-86 THESIS',/,11X,
1 'DC SEED D(I)=PKSTOR(3,I)
100 SEED D(I)=PKSTOR(3,I)
101 CALL PLCTP (SEEABS,SEED,NN,0)
102 IF (LOOK(8).EQ.0) GO TO 98
103 WRITE (6,96)
104 FORMAT (1H1,10X,'KETRON MK-86 THESIS',/,11X,
1 'DC SEED D(I)=RMEAN(I,4)
105 SEED D(I)=RMEAN(I,4)
106 CALL PLCTP (SEEABS,SEED,NN,0)
107 IF (LOOK(9).EQ.0) GO TO 101
108 WRITE (6,99)
109 FORMAT (1H1,10X,'KETRON MK-86 THESIS',/,11X,
1 'DC SEED D(I)=RMEAN(I,5)
110 SEED D(I)=RMEAN(I,5)
111 CALL PLCTP (SEEABS,SEED,NN,0)
112 IF (LOOK(10).EQ.0) GO TO 104
113 WRITE (6,102)
114 FORMAT (1H1,10X,'KETRON MK-86 THESIS',/,11X,
1 'DC SEED D(I)=RMEAN(I,6)
115 SEED D(I)=RMEAN(I,6)
116 CALL PLCTP (SEEABS,SEED,NN,0)
117 IF (LOOK(11).EQ.0) GO TO 107

```

MCSP0622
MCSP0623
MCSP0624
MCSP0625
MCSP0626
MCSP0627
MCSP0628
MCSP0629
MCSP0630
MCSP0631
MCSP0632
MCSP0633
MCSP0634
MCSP0635
MCSP0636
MCSP0637
MCSP0638
MCSP0639
MCSP0640
MCSP0641
MCSP0642
MCSP0643
MCSP0644
MCSP0645
MCSP0646
MCSP0647
MCSP0648
MCSP0649
MCSP0650
MCSP0651
MCSP0652
MCSP0653
MCSP0654
MCSP0655
MCSP0656
MCSP0657
MCSP0658
MCSP0659
MCSP0660
MCSP0661
MCSP0662
MCSP0663
MCSP0664
MCSP0665
MCSP0666
MCSP0667
MCSP0668
MCSP0669


```

105 WRITE (6,105)
106 FORMAT (I1,I10X,'KETRON MK-86 THESIS',/,11X,
107 'X VELOCITY VARIANCE',//)
108 DO 106 I=1,NN
109 SEORD(I)=PKSTOR(2,I)
110 CALL PLOTP (SEEABS,SEORD,NN,0)
111 IF (LOOK(12).EQ.0) GO TO 110
112 WRITE (6,108)
113 FORMAT (I1,I10X,'KETRON MK-86 THESIS',/,11X,
114 'Y VELOCITY VARIANCE',//)
115 DO 109 I=1,NN
116 SEORD(I)=PKSTOR(N4,I)
117 CALL PLOTP (SEEABS,SEORD,NN,0)
118 IF (LOOK(13).EQ.0) GO TO 113
119 WRITE (6,111)
120 FORMAT (I1,I10X,'KETRON MK-86 THESIS',/,11X,
121 'Z VELOCITY VARIANCE',//)
122 DO 112 I=1,NN
123 SEORD(I)=PKSTOR(N5,I)
124 CALL PLOTP (SEEABS,SEORD,NN,0)
125 IF (LOOK(14).EQ.0) GO TO 116
126 WRITE (6,114)
127 FORMAT (I1,I10X,'KETRON MK-86 THESIS',/,11X,
128 'TARGET SPEED IN FT/SEC',//)
129 DO 115 I=1,NN
130 SEORD(I)=DSQRT((RMEAN(I,4)+TVELX(I))**2+(RMEAN(I,5)+TVELY(I))**2+
131 (RMEAN(I,6)+TVELZ(I))**2)*3.D0
132 CALL PLOTP (SEEABS,SEORD,NN,0)
133 CONTINUE
134
135 THE FOLLOWING SEGMENT PUNCHES GAIN MATRICES IF REQUESTED
136
137 IF (IG.NE.1) GO TO 119
138 DO 117 I=1,NN
139 DO 117 J=1,N
140 WRITE (7,118) I,J,(GSTOR(J,JJ,I),JJ=1,3)
141 FORMAT (3X,2I4,3D20.8)
142 STOP
143 FCPRMAT(1H0,/,40X,'AVERAGE FILTER VELOCITY ESTIMATE ERROR = ',
144 D20.8,' YARDS/SEC.')
145 FCPRMAT(1H0,/,,' GAMMA MATRIX',//)
146 END

```

SUBROUTINE NOISE (RT,BT,ET,R1,RB,RE,KERNL1,KERNL2,KERNL3,X,Y,Z, MCSP0712 MCSP0711


```

C C C
1 INDEX)
  THIS SUBROUTINE TAKES THE INPUT R,B,E DATA, ADDS NOISE, AND THEN
  CONVERTS INTO NOISY X,Y,Z.
  IMPLICIT REAL*8(A-H,L-Z)
  REAL*4 RERR,BERR,EERR
  RERROR = 0
  BERROR = 0
  EERROR = 0
  IF (INDEX.EQ.0) GO TO 1
    SET RANGE STANDARD DEVIATION AT INPUT VALUE + .1 PER CENT RANGE
    RR=PI+.001DO*RT
    CALL NORMAL (KERNL1,RERR,1)
    RERROR=RR*DBLE(RERR)
    CALL NORMAL (KERNL2,BERR,1)
    BERROR=RB*DBLE(BERR)
    CALL NORMAL (KERNL3,EERR,1)
    EERROR=RE*DBLE(EERR)
    K=RT+RERROR
    B=BT+BERROR
    E=ET+EERROR
  1
  CONVERT INTO NOISY X,Y,Z DATA
  X=R*DCOS(B)*DCOS(E)
  Y=R*DSIN(B)*DCOS(E)
  Z=R*DSIN(E)
  RETURN
END
C C C

```

```

C C C C C C
SUBROUTINE UPDATE (N,NUM,RADAR,STATES,PRED,IFLAG,L,TIME,IGAIN,ITER
1,K1,K2,ADAPTQ)
  THIS ROUTINE COMPUTES THE FILTER ESTIMATE X(K/K)
  AND FILTER PREDICTION X(K+1/K)
  IT ALSO GENERATES THE SYNTHETIC ESTIMATES IF REQUESTED, AND
  THE ADAPTIVE Q MATRIX IS UPDATED IF DESIRED
  IMPLICIT REAL*8(P,Q,H,G,R,I,S)
  REAL*8 TEMPI,TEMP2,TEMP3,II,AHEAD,EST,K1,K2
  INTEGER*4 ADAPTQ
  DIMENSION RADAR(9,9), STATES(9,9), PRED(9,9), TEMPI(9,9), TEMP2(9,MCSP0744
MCSP0745
MCSP0746
MCSP0747
MCSP0748
MCSP0749
MCSP0750
MCSP0751
MCSP0752
MCSP0753
MCSP0754
MCSP0755

```



```

19), TEMP3(9,9), QOLD(9,9), QRESID(9,9), SPDIF(9,9), SPTR(9,9)
CCMMCN II(9,9), Q(9,9), H(9,9), R(9,9), G(9,9), PHI(9,9), PKK(9,9), PHIPK
1(S,9), RMEAN(233,6), RSD(6), RESID(233,4), AHEAD(233), EST(9,233),
2M(9,9,40), PKSTOR(9,233)
DC I I=1,9
DC J J=1,9
TEMP1(I,J)=0.00
TEMP2(I,J)=0.00
TEMP3(I,J)=0.00
1 IF(IFLAG.EQ.1) GO TO 2
      X(K/K) = X(K/K-1) + G(K)(Z(K)-HX(K/K-1))
      CALL PRCD (H,PRED,NUM,N,1,TEMP1)
      CALL SUB (RADAR,TEMP1,NUM,1,TEMP3)
      CALL PRCD (G,TEMP3,N,NUM,1,TEMP2)
      CALL ADD (PRED,TEMP2,N,1,STATES)
      IFLAG = 1 SIGNIFIES THAT THE ESTIMATED POSITIONS ARE TO BE THE
      MEASURED POSITIONS - THIS OPTION IS ONLY AVAILABLE FOR
      THE CASES WHERE GAINS ARE CALCULATED BY THE PROGRAM
2 IF(IFLAG.NE.1.OR.IGAIN.EQ.2) GO TO 5
  CALL TRANS (PHI,N,N,TEMP2)
      THE FOLLOWING SEGMENT GENERATES SYNTHETIC FILTER ESTIMATES AND
      COVARIANCES OF ESTIMATION ERROR IF REQUESTED BY 'OPT' = 1
  N2=N/3+1
  N3=N*2/3+1
  N4=N+1
  N5=N3+1
  STATES(1,1)=RADAR(1,1)
  STATES(N2,1)=RADAR(2,1)
  STATES(N3,1)=RADAR(3,1)
  IF (LSEQ.1) GO TO 5
  STATES(N4,1)=(RADAR(1,1)-EST(1,L-1))/TIME
  STATES(N5,1)=(RADAR(2,1)-EST(N2,L-1))/TIME
  STATES(N5,1)=(RADAR(3,1)-EST(N3,L-1))/TIME
  PKK(1,1)=1.00
  PKK(N2,N2)=1.00
  PKK(N3,N3)=1.00
  PKK(1,2)=1.00/TIME
  PKK(N2,N4)=PKK(1,2)
  PKK(N3,N5)=PKK(1,2)
  PKK(2,1)=1.00/TIME
  PKK(N4,N2)=PKK(2,1)
  PKK(N5,N3)=PKK(2,1)

```



```

C C C C C
PKK(2,2)=2.DO/TIME**2
PKK(N4,N4)=PKK(2,2)
PKK(N5,N5)=PKK(2,2)

IF FILTER IS 2ND ORDER OR 3 SAMPLES HAVE NOT BEEN TAKEN TO
CALCULATE FINAL 3RD ORDER ESTIMATES AND COVARIANCE TERMS,
SKIP THE FOLLOWING SECTION

IF (N.EQ.6) GO TO 3
IF (N.EQ.9.AND.L.LT.3) GO TO 5
N6=N4+1
N7=N5+1
STATES(N6,1)=(RADAR(1,1)-2.DO*EST(1,L-1)+EST(1,L-2))/TIME**2
STATES(N7,1)=(RADAR(2,1)-2.DO*EST(N2,L-1)+EST(N2,L-2))/TIME**2
STATES(N3,1)=(RADAR(3,1)-2.DO*EST(N3,L-1)+EST(N3,L-2))/TIME**2
PKK(1,3)=1.DO/TIME**2
PKK(N2,N6)=PKK(1,3)
PKK(N3,N7)=PKK(1,3)
PKK(2,3)=3.DO/TIME**3
PKK(N4,N6)=PKK(2,3)
PKK(N5,N7)=PKK(2,3)
PKK(3,1)=1.DO/TIME**2
PKK(N6,N2)=PKK(3,1)
PKK(N7,N3)=PKK(3,1)
PKK(3,2)=3.DO/TIME**3
PKK(N6,N4)=PKK(3,2)
PKK(N7,N5)=PKK(3,2)
PKK(3,3)=6.DO/TIME**4
PKK(N6,N6)=PKK(3,3)
PKK(N7,N7)=PKK(3,3)
DO 4 I=1,N
DO 4 J=1,N
PKKM1(I,J)=0.DO
CALL PRCD (PHI,PKK,N,N,TEMP1)
CALL PRCD (TEMP1,TEMP2,N,N,TEMP3)
CALL ADD(TEMP3,Q,N,N,PKKM1)
CONTINUE

THE FOLLOWING SEGMENT IS USED TO UPDATE THE ADAPTIVE Q MATRIX
IF (ADAPTQ.NE.1.OR.IGAIN.EQ.2.OR.(IGAIN.EQ.1.AND.ITER.NE.1)) GO TO 6
DO 6 I=1,N
DO 6 J=1,N
STATES(I,J)=STATES(I,J)-PRED(I,J)
SPDIFF(N,N,SPTR)
CALL TRANS (SPDIFF,SPTR,N,N,N,QRESID)
DO 7 I=1,N

```

MCSP0804
 MCSP0805
 MCSP0806
 MCSP0807
 MCSP0808
 MCSP0809
 MCSP0810
 MCSP0811
 MCSP0812
 MCSP0813
 MCSP0814
 MCSP0815
 MCSP0816
 MCSP0817
 MCSP0818
 MCSP0819
 MCSP0820
 MCSP0821
 MCSP0822
 MCSP0823
 MCSP0824
 MCSP0825
 MCSP0826
 MCSP0827
 MCSP0828
 MCSP0829
 MCSP0830
 MCSP0831
 MCSP0832
 MCSP0833
 MCSP0834
 MCSP0835
 MCSP0836
 MCSP0837
 MCSP0838
 MCSP0839
 MCSP0840
 MCSP0841
 MCSP0842
 MCSP0843
 MCSP0844
 MCSP0845
 MCSP0846
 MCSP0847
 MCSP0848
 MCSP0849
 MCSP0850
 MCSP0851

MCSP08552
MCSP08553
MCSP08554
MCSP08555
MCSP08556
MCSP08557
MCSP08558
MCSP08559
MCSP08560
MCSP08561

```

DC 7 J=1,N
QOLD(I,J)=Q(I,J)
Q(I,J) = K1*QOLD(I,J) + K2*QRESID(I,J)
7 CCNTINUE
      X(K+1/K) = PHI * X(K/K)
8 CALL PROCD (PHI,STATES,N,N,1,PRED)
  RETURN
  END

```

CC C

SUBROUTINE SCENAR (X,Y,Z,STATES,N,L,ITER,IGAIN,TVELX,TVELY,TVELZ,LEVEL)

MCSP08662
MCSP08663
MCSP08664
MCSP08665
MCSP08666
MCSP08667
MCSP08668
MCSP08669
MCSP08670
MCSP08671
MCSP08672
MCSP08673
MCSP08674
MCSP08675
MCSP08676
MCSP08677
MCSP08678
MCSP08679
MCSP08680
MCSP08681
MCSP08682
MCSP08683
MCSP08684
MCSP08685
MCSP08686
MCSP08687
MCSP08688
MCSP08689
MCSP08690
MCSP08691
MCSP08692
MCSP08693
MCSP08694

THIS SUBROUTINE COMPUTES THE MEANS OF THE FILTER ESTIMATES
FOR ITER! SAMPLES FOR USE IN CALCULATING THE FILTER
PERFORMANCE FACTOR AND ALSO THE VARIANCES WHEN GAINS ARE READ

```

IMPLICIT REAL*8(P,Q,H,G,R,I,S)
REAL*8 X,Y,Z,XCAL,YCAL,ZCAL,TERM1,TERM2,II,AHEAD,EST
DIMENSION II(9,9),STATES(9,9),H(9,9),R(9,9),G(9,9),PHI(9,9),PKKM1(9,9),RMEAN(233,6),RSD(6),AHEADC(233,4),EST(9,233),PHIPR(9,9),RMEAN(9,233),PKSTOR(9,233)
1(9,9),RMEAN(233,6),RSD(6),AHEADC(233,4),EST(9,233),PHIPR(9,9),RMEAN(9,233),PKSTOR(9,233)
2N(9,9,40),PKSTOR(9,233)
N2=N/3+1
N3=N*2/3+1
N4=N2+1
N5=N3+1
XCAL=STATES(1,1)
YCAL=STATES(N2,1)
ZCAL=STATES(N3,1)

```

CCCCC

RMEAN'S ARE SUM OF FILTER ESTIMATES IN THIS ROUTINE - AFTER FULL
MCNTE CARLO RUN HAS BEEN CONCLUDED THE MEANS ARE CALCULATED BY
DIVIDING THESE SUMS BY THE NUMBER OF MONTE CARLO RUNS IN MAIN

```

RMEAN(L,1)=RMEAN(L,1)+XCAL
RMEAN(L,2)=RMEAN(L,2)+YCAL
RMEAN(L,3)=RMEAN(L,3)+ZCAL
RMEAN(L,4)=RMEAN(L,4)+STATES(2,1)
RMEAN(L,5)=RMEAN(L,5)+STATES(N4,1)
RMEAN(L,6)=RMEAN(L,6)+STATES(N5,1)

```

CCCCC

SIMILARLY, PKSTOR'S ARE THE SUMS OF SQUARES OF ESTIMATION ERROR

CC


```

CC      AT EACH POINT
      PKSTOR(1,L)=PKSTOR(1,L)+(XCAL-X)**2
      PKSTOR(N2,L)=PKSTOR(N2,L)+(YCAL-Y)**2
      PKSTOR(N3,L)=PKSTOR(N3,L)+(ZCAL-Z)**2
      IF (KVEL.EQ.1) GO TO 1
      IF TRUE VELOCITIES ARE NOT AVAILABLE THE VARIANCE ABOUT THE
      ESTIMATE, NOT TRUTH, WILL BE CALCULATED USING THE NEXT 3 CARDS
      PKSTOR(2,L)=PKSTOR(2,L)+STATES(2,1)**2
      PKSTOR(N4,L)=PKSTOR(N4,L)+STATES(N4,1)**2
      PKSTOR(N5,L)=PKSTOR(N5,L)+STATES(N5,1)**2
      RETURN
1      PKSTOR(2,L)=PKSTOR(2,L)+(STATES(2,1)-TVELX)**2
      PKSTOR(N4,L)=PKSTOR(N4,L)+(STATES(N4,1)-TVELY)**2
      PKSTOR(N5,L)=PKSTOR(N5,L)+(STATES(N5,1)-TVELZ)**2
      RETURN
      END

```

MCSP0895
 MCSP0896
 MCSP0897
 MCSP0898
 MCSP0899
 MCSP0900
 MCSP0901
 MCSP0902
 MCSP0903
 MCSP0904
 MCSP0905
 MCSP0906
 MCSP0907
 MCSP0908
 MCSP0909
 MCSP0910
 MCSP0911
 MCSP0912
 MCSP0913

```

CC      SUBROUTINE GAIN (N,M,ITER,L,IFLAG,KVEL)
CC      THIS SUBROUTINE COMPUTES THE OPTIMUM GAIN MATRIX AND THE ERRGR
CC      COVARIANCE
      IMPLICIT REAL*8(P,Q,H,G,R,T)
      REAL*8 II,AHEAD,EST
      DIMENSION TEMP(9,9), TEMP1(9,9), PHIT(9,9), HT(9,9), T
1      TEMP2(9,9), TEMP4(9,9)
      COMMON II(9,9),Q(9,9),H(9,9),R(9,9),G(9,9),PHI(9,9),PKK(9,9),PKKM1
1      (9,9),RMEAN(233,6),RSD(6),RESID(233,4),AHEAD(233),EST(9,233),PHIPRM
2      M(9,9,40),PKSTOR(9,233)
      THE FOLLOWING CARD PRECLUDES THE CALCULATION OF GAINS IF THE
      SYNTHETIC ESTIMATION INITIALIZATION IS USED
      IF (IFLAG.EQ.1) RETURN
      G(K) = P(K/K-1)*HT*(H*P(K/K-1)*HT + R)
      DO 1 I=1,9
      DO 1 J=1,9

```

MCSP0914
 MCSP0915
 MCSP0916
 MCSP0917
 MCSP0918
 MCSP0919
 MCSP0920
 MCSP0921
 MCSP0922
 MCSP0923
 MCSP0924
 MCSP0925
 MCSP0926
 MCSP0927
 MCSP0928
 MCSP0929
 MCSP0930
 MCSP0931
 MCSP0932
 MCSP0933
 MCSP0934
 MCSP0935
 MCSP0936
 MCSP0937

MCSP0938
MCSP0939
MCSP0940
MCSP0941
MCSP0942
MCSP0943
MCSP0944
MCSP0945
MCSP0946
MCSP0947
MCSP0948
MCSP0949
MCSP0950
MCSP0951
MCSP0952
MCSP0953
MCSP0954
MCSP0955
MCSP0956
MCSP0957
MCSP0958
MCSP0959
MCSP0960
MCSP0961
MCSP0965
MCSP0966
MCSP0967
MCSP0968
MCSP0969
MCSP0970
MCSP0971
MCSP0972
MCSP0973
MCSP0974
MCSP0975
MCSP0976
MCSP0977

MCSP0978
MCSP0979
MCSP0980
MCSP0981
MCSP0982
MCSP0983

```

PHIT(I,J)=0.D0
HT(I,J)=0.D0
TEMP1(I,J)=0.D0
TEMP2(I,J)=0.D0
TEMP3(I,J)=0.D0
TEMP4(I,J)=0.D0
CALL TRANS (H,M,N,HT)
CALL PRCD (PKKM1,HT,N,M,TEMP)
CALL PRCD (H,TEMP,M,N,M,TEMP1)
CALL ADD (TEMP1,R,M,M,TEMP3)
IF (N.EQ.1) GO TO 5
MD=9
CALL GAUSS3 (3,EPS1,TEMP3,TEMP2,KER,MD)
CALL PRCD (TEMP,TEMP2,N,M,M,G)

NOTE HERE PKK(I,J) = P(K/K) WHERE
P(K/K) = (I-G(K)*H)*P(K/K-1)

2 CALL PRCD (G,H,N,M,N,TEMP3)
CALL SUB (II,TEMP3,N,N,TEMP4)
CALL PRCD (TEMP4,PKKM1,N,N,N,PKK)

MCSP SEQUENCE NUMBERS INTERRUPTED

NOTE HERE PKKM1(I,J) = P(K/K-1) WHERE
P(K/K-1) = PHI*P(K-1/K-1)*PHIT + Q

4 CALL TRANS (PHI,N,N,PHIT)
CALL PRCD (PKK,PHIT,N,N,N,TEMP3)
CALL PRCD (PHI,TEMP3,N,N,N,TEMP4)
CALL ADD (TEMP4,Q,N,N,PKKM1)
RETURN
DC 6 I=1,9
5 G(I,1)=TEMP(1,1)/TEMP1(1,1)
6 GO TO 2
END

```

CCCC

CCCCCCC

SUBROUTINE ADD (A,B,N,M,C)
THIS SUBROUTINE ADDS THE NXM MATRICES A AND B, STORING THE
RESULT IN C
REAL*8 A,B,C

CCCC


```

DIMENSION A(9,9), B(9,9), C(9,9)
DO 1 I=1,N
DO 1 J=1,M
1 C(I,J)=A(I,J)+B(I,J)
RETURN
END

```

MCSPI0984
 MCSPI0985
 MCSPI0986
 MCSPI0987
 MCSPI0988
 MCSPI0989

```

C
C
C
SUBROUTINE SUB (A,B,N,M,C)
  THIS SUBROUTINE SUBTRACTS THE NXM MATRIX B FROM THE NXM MATRIX
  A AND STORES THE RESULT IN C
  REAL*8 A,B,C
  DIMENSION A(9,9), B(9,9), C(9,9)
  DO 1 I=1,N
  DO 1 J=1,M
  1 C(I,J)=A(I,J)-B(I,J)
  RETURN
END

```

MCSPI0990
 MCSPI0991
 MCSPI0992
 MCSPI0993
 MCSPI0994
 MCSPI0995
 MCSPI0996
 MCSPI0997
 MCSPI0998
 MCSPI0999
 MCSPI1000
 MCSPI1001

```

C
C
C
C
SUBROUTINE PROD (A,B,N,M,L,C)
  THIS SUBROUTINE COMPUTES THE MATRIX PRODUCT AB AND STORES THE
  RESULT IN C
  A = NXM    B = MXL    C = NXL
  REAL*8 A,B,C
  DIMENSION A(9,9), B(9,9), C(9,9)
  DO 1 I=1,N
  DO 1 J=1,L
  1 C(I,J)=0.0
  DO 2 I=1,N
  DO 2 J=1,L
  DO 2 K=1,M
  2 C(I,J)=C(I,J)+A(I,K)*B(K,J)
  RETURN
END

```

MCSPI1002
 MCSPI1003
 MCSPI1004
 MCSPI1005
 MCSPI1006
 MCSPI1007
 MCSPI1008
 MCSPI1009
 MCSPI1010
 MCSPI1011
 MCSPI1012
 MCSPI1013
 MCSPI1014
 MCSPI1015
 MCSPI1016
 MCSPI1017
 MCSPI1018

MCSPI019
 MCSPI020
 MCSPI021
 MCSPI022
 MCSPI023
 MCSPI024
 MCSPI025
 MCSPI026
 MCSPI027
 MCSPI028
 MCSPI029
 MCSPI030
 MCSPI031

```

SUBROUTINE TRANS (A,N,M,C)
  THIS SUBROUTINE FORMS THE MATRIX TRANSPOSE OF A STORING THE
  RESULT IN C
  A = NXM
  C = MXN
  REAL*8 A,C
  DIMENSION A(9,9), C(9,9)
  DO 1 I=1,N
  DO 1 J=1,M
    1 C(J,I)=A(I,J)
  RETURN
END
  
```

MCSPI032
 MCSPI033
 MCSPI034
 MCSPI035
 MCSPI036
 MCSPI037
 MCSPI038
 MCSPI039
 MCSPI040
 MCSPI041
 MCSPI042
 MCSPI043
 MCSPI044

```

SUBROUTINE MREAD (A,N,M)
  THIS SUBROUTINE READS AN NXM MATRIX A ACCORDING TO THE FORMAT
  8D10.5. THE ENTRIES IN THE FIRST ROW OF A ARE READ FIRST, THEN
  THE ENTRIES IN THE SECOND ROW, ETC.
  REAL*8 A
  DIMENSION A(9,9)
  DO 1 I=1,N
    1 READ (5,2) (A(I,J),J=1,M)
  FORMAT (8D10.5)
  RETURN
END
  
```

MCSPI045
 MCSPI046
 MCSPI047
 MCSPI048
 MCSPI049
 MCSPI050
 MCSPI051
 MCSPI052
 MCSPI053
 MCSPI054
 MCSPI055

```

SUBROUTINE MREAD1 (A,N,M)
  THIS SUBROUTINE READS AN NXM MATRIX A ACCORDING TO THE FORMAT
  10X,3D20.8). THE ENTRIES IN THE FIRST ROW OF A ARE READ FIRST,
  THE ENTRIES IN THE SECOND ROW, ETC.
  REAL*8 A
  DIMENSION A(9,9)
  DO 1 I=1,N
    1 READ (5,2) (A(I,J),J=1,M)
  FORMAT (10X,3D20.8)
  RETURN
  
```


END

MCSP1057

```

C
C
C
SUBROUTINE MWRITE (A,N,M)
  THIS SUBROUTINE WRITES THE ENTRIES OF THE NXM MATRIX A
  REAL*8 A
  DIMENSION A(9,9)
  DO 1 I=1,N
    DO 2 J=1,M
      WRITE(6,2) (I,J,A(I,J),J=1,M)
      FORMAT(2(3X,I,II,II,II)=',1PD19.12))
    1 RETURN
  2
  END

```

MCSP1058
MCSP1059
MCSP1060
MCSP1061
MCSP1062
MCSP1063
MCSP1064
MCSP1065
MCSP1066
MCSP1067
MCSP1068

```

C
C
C
SUBROUTINE GAUSS3 (N,EPS1,A,X,KER,K)
  THIS SUBROUTINE INVERTS AN N X N MATRIX THAT HAS BEEN
  DIMENSIONED K X K IN THE CALLING PROGRAM
  REAL*8 A,X,Y,D
  DIMENSION A(1), X(1), L(9), M(9), Y(9,9)
  DO 1 I=1,N
    DO 2 J=1,N
      IND=(I-1)*K+J
      Y(I,J)=A(IND)
    1 KER=1
    N2=2*N
    CALL ARRAY (2,N,N,9,9,Y,Y)
    CALL DMINV (Y,N,D,L,M)
    CALL ARRAY (1,N,N,9,9,Y,Y)
    IF (D.EQ.0) KER=2
    DO 2 I=1,N
      DO 2 J=1,N
        IND=(I-1)*K+J
        X(IND)=Y(I,J)
      2
    END

```

MCSP1069
MCSP1070
MCSP1071
MCSP1072
MCSP1073
MCSP1074
MCSP1075
MCSP1076
MCSP1077
MCSP1078
MCSP1079
MCSP1080
MCSP1081
MCSP1082
MCSP1083
MCSP1084
MCSP1085
MCSP1086
MCSP1087
MCSP1088
MCSP1089
MCSP1090
MCSP1091


```

SUBROUTINE ARRAY (MODE,I,J,N,M,S,D)
DIMENSION S(1), D(1)
REAL*8 S,D
NI=N-I
IF (MODE-1) 1,1,3
1 IJ=I*J+1
NM=N*J+1
DC 2 K=1,J
NM=NM-NI
DC 2 L=1,I
IJ=IJ-1
NM=NM-1
2 D(NM)=S(IJ)
3 IJ=0
NM=0
DC 5 K=1,J
DC 4 L=1,I
IJ=IJ+1
NM=NM+1
4 S(IJ)=D(NM)
5 NM=NM+NI
6 RETURN
END

```

MCSPI1092
 MCSPI1093
 MCSPI1094
 MCSPI1095
 MCSPI1096
 MCSPI1097
 MCSPI1098
 MCSPI1099
 MCSPI1100
 MCSPI1101
 MCSPI1102
 MCSPI1103
 MCSPI1104
 MCSPI1105
 MCSPI1106
 MCSPI1107
 MCSPI1108
 MCSPI1109
 MCSPI1110
 MCSPI1111
 MCSPI1112
 MCSPI1113
 MCSPI1114
 MCSPI1115

```

SUBROUTINE RNOISE (XM,RB1,RRO,RE1,N)
THIS SUBROUTINE CALCULATES THE COVARIANCE CF MEASUREMENT NOISE
MATRIX, R, FOR ONLINE CALCULATION IF REQUESTED
RB1,RRO,RE1 ARE INPUT AS STANDARD DEVIATIONS
IMPLICIT REAL*8(P,Q,H,G,R,T,B)
REAL*8 I,X,DUMMY,RAN,BEAR,ELEV,SINB,SINE,CCSB,COSE,SSQB,SSQE,CSQB
1,CSQEQ,RSQ,XM,AHEAD,EST
COMMON I(9,9),Q(9,9),H(9,9),R(9,9),G(9,9),PHI(9,9),PKKM1
1(9,9),KMEAN(233,6),RSD(6),RESID(233,4),AHEAD(233),EST(9,233),PHIPR
2M(9,9,40),PKSTOR(9,233)
DIMENSION X(9), XM(9,9)
DC 1 I=1,9
1 X(I)=XM(I,1)
PI=3.14159265358979323846
N2=N/5+1

```

MCSPI1116
 MCSPI1117
 MCSPI1118
 MCSPI1119
 MCSPI1120
 MCSPI1121
 MCSPI1122
 MCSPI1123
 MCSPI1124
 MCSPI1125
 MCSPI1126
 MCSPI1127
 MCSPI1128
 MCSPI1129
 MCSPI1130
 MCSPI1131
 MCSPI1132
 MCSPI1133
 MCSPI1134

CCCCCCCC


```

N3=N*2/3+1
N3=DSQRT(X(1)**2+X(N2)**2+X(N3)**2)
RR1=RR0+.001D0*RRAN
IF (X(1)).EQ.0.0) GO TO 2
IF (X(N2)).EQ.0.0D0.AND.X(1).GT.0.0D0) BEAR=0.0
IF (X(N2)).EQ.0.0D0.AND.X(1).LT.0.0D0) BEAR=PI
IF (X(N2)).NE.0.0D0) BEAR=DATAN(X(N2)/X(1))
GO TO 3
2 IF (X(N2)).GT.0.0) BEAR = PI/2.0
IF (X(N2)).LT.0.0) BEAR=3.*PI/2.
3 DUMMY=X(N3)/RRAN
GO TO 5
4 BEAR=0.0D0
DUMMY=1.0D0
5 IF (DUMMY.NE.1.0) ELEV=PI/2.0
IF (DUMMY.NE.1.0) ELEV=DATAN(DUMMY/DSQRT(1.-DUMMY**2))
SINB=DSIN(BEAR)
COSB=DCOS(BEAR)
SSQB=SINB**SINE
SSQB=COSB**COSE
SSQB=RR1*RR1
SSQB=RR1*RR1
RE=RE1*RR1
RE(1,1)=RSQ*RE*CSQB*SSQB+RSQ*RB*SSQB*CSQB+RSQ*RB*RE*SSQB*SSQB+RR*CS
1 QB*CSQB
RR(1,2)=RSQ*CSQB*SINB*SSQB*(RE-RB*RE)+SINB*CCSB*CSQB*(RR-RSQ*RB)
RR(2,1)=RSQ*RE*SSQB*SSQB+RSQ*RB*CSQB*CSQB+RSQ*RB*RE*CSQB*SSQB+RR*SS
1 QB*CSQB
RR(2,3)=SINB*SINE*COSE*(RR-RSQ*RE)
RR(3,2)=R(2,3)=SINB*COSE*(RR-RSQ*RE)
RR(3,1)=R(1,3)=R(1,3)
RR(3,3)=RSQ*CSQB*RE+SSQB*RR
END

```

MCSPI1135
 MCSPI1136
 MCSPI1137
 MCSPI1138
 MCSPI1139
 MCSPI1140
 MCSPI1141
 MCSPI1142
 MCSPI1143
 MCSPI1144
 MCSPI1145
 MCSPI1146
 MCSPI1147
 MCSPI1148
 MCSPI1149
 MCSPI1150
 MCSPI1151
 MCSPI1152
 MCSPI1153
 MCSPI1154
 MCSPI1155
 MCSPI1156
 MCSPI1157
 MCSPI1158
 MCSPI1159
 MCSPI1160
 MCSPI1161
 MCSPI1162
 MCSPI1163
 MCSPI1164
 MCSPI1165
 MCSPI1166
 MCSPI1167
 MCSPI1168
 MCSPI1169
 MCSPI1170
 MCSPI1171
 MCSPI1172
 MCSPI1173
 MCSPI1174
 MCSPI1175
 MCSPI1176

MCSPI1177

SUBROUTINE ACCUR (X,Y,Z,L,N,ITER)

CCCC

```

      THIS SUBROUTINE CALCULATES THE SHELL AT TARGET RESIDUALS IF
      REQUESTED
      IMPLICIT REAL*8(Q,H,R,G,P,A,B,T,S,X,Y,Z,E)
      REAL*8 DIFF,II
      DIMENSION A(9,9),TEMP(9,9),TRY(9,9)
      COMMON II(9,9),Q(9,9),H(9,9),R(9,9),G(9,9),PHI(9,9),PKK(9,9),PKKMI
      1(9,9),RMEAN(233,6),RSD(6),RESID(233,4),AHEAD(233),EST(9,233),PHIPRM
      2M(9,9,40),PKSTOR(9,233)
      DIFF=FLOAT(L)-4.00*AHEAD(L)
      LESS=DIFF
      DO 2 I=1,9
      DO 3 J=2,9
      1 A(I,J)=EST(I,LESS)
      2 IAHEAD=AHEAD(L)*4.00+1.00
      DO 3 I=1,9
      DO 3 J=1,9
      3 TEMP(I,J)=PHIPRM(I,J,IAHEAD)
      CALL PRGD (TEMP,A,N,N,1,TRY)
      NZ=N/3+1
      NC=N*2/3+1
      XCAL=TRY(1,1)
      YCAL=TRY(N2,1)
      ZCAL=TRY(N3,1)
      RESID(L,1)=RESID(L,1)+(XCAL-X)
      RESID(L,2)=RESID(L,2)+(YCAL-Y)
      RESID(L,3)=RESID(L,3)+(ZCAL-Z)
      RETURN
      END

```

MCSP11178
MCSP11179
MCSP11180
MCSP11181
MCSP11182
MCSP11183
MCSP11184
MCSP11185
MCSP11186
MCSP11187
MCSP11188
MCSP11189
MCSP11190
MCSP11191
MCSP11192
MCSP11193
MCSP11194
MCSP11195
MCSP11196
MCSP11197
MCSP11198
MCSP11199
MCSP1200
MCSP1201
MCSP1202
MCSP1203
MCSP1204
MCSP1205
MCSP1206
MCSP1207
MCSP1208

SUBROUTINE STUDY (I,X,Y,Z,RESIDX,RESIDY,RESIDZ,AHEAD,RMISS)

THIS ROUTINE OUTPUTS THE SHELL AT TARGET RESIDUAL
PERFORMANCE TABLE IF REQUESTED

```

      REAL*8 X,Y,Z,RESIDX,RESIDY,RESIDZ,CALX,CALY,CALZ,AHEAD,TIME,RMISS
      CALX=X+RESIDX
      CALY=Y+RESIDY
      CALZ=Z+RESIDZ
      TIME=FLOAT(I)-AHEAD*4.00
      WRITE (6,1) I,X,CALX,RESIDX,AHEAD
      WRITE (6,2) Y,CALY,RESIDY,RMISS

```

MCSP1209
MCSP1210
MCSP1211
MCSP1212
MCSP1213
MCSP1214
MCSP1215
MCSP1216
MCSP1217
MCSP1218
MCSP1219
MCSP1220


```

1  WRITE (6,3) Z,CALZ,RESIDZ,TIME
2  FCRMAT (1H0,14,X=,3D15.6,D14.5)
3  FCRMAT (1H,5X,Y=,3D15.6,-----,D15.6)
   FCRMAT (1H,5X,Z=,3D15.6,D14.5,/)
   RETURN
   END

```

```

MCSP1221
MCSP1222
MCSP1223
MCSP1224
MCSP1225
MCSP1226

```


APPENDIX D

LISTING OF SHELL FLIGHT TIME INTERPOLATION PROGRAM

The program that interpolates in the 5 in - 54 gunfire table (included after the program listing) to generate the shell-at-target flight times is included in this appendix. In its present form it can provide times for tracks operating in a region bounded by

minimum range	- 500 yards
maximum range	-7500 yards
minimum elevation angle	- 15 degrees
minimum elevation angle	- 90 degrees.

If track data points fall outside of these limits the program can be easily expanded by enlarging the arrays and revising the linear interpolation equations.


```

C      FRAC = (R-(INDR-1)*500.DO)/500.DO
C      T1 = ARRAY(INDR-1, INDE-1) + FRAC*(ARRAY(INDR, INDE-1) - ARRAY(INDR-1,
1  INDE-1))
C      T2 = ARRAY(INDR-1, INDE) + FRAC*(ARRAY(INDR, INDE) - ARRAY(INDR-1, INDE))
C
C      THE FLIGHT TIME IS CALCULATED IN SECS
C
C      TIME = T1 + (E-ESTEP1 )/5.DO*(T2-T1)
C      WRITE(6,51) I,X,Y,Z,R,E,TIME
C      FORMAT(IH0,I3,6D15.6,5X,'-----',10X,'-----')
C      CCNTINUE
C      51
C      1000 STOP
C      END

```


5 INCH 54 CALIBER GUN I.V. 2500 F.S.

TIME OF FLIGHT IN SECONDS

SLANT RANGE YARDS	POSITION ANGLE IN DEGREES																		
	0	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90
500	0.61	0.61	0.61	0.61	0.61	0.61	0.61	0.61	0.61	0.61	0.61	0.61	0.61	0.61	0.61	0.61	0.61	0.61	0.61
1000	1.24	1.24	1.24	1.24	1.24	1.24	1.24	1.24	1.24	1.24	1.24	1.24	1.24	1.24	1.24	1.24	1.24	1.25	1.25
1500	1.88	1.88	1.89	1.89	1.89	1.89	1.89	1.89	1.90	1.90	1.90	1.90	1.90	1.90	1.90	1.90	1.90	1.90	1.90
2000	2.55	2.55	2.56	2.56	2.56	2.56	2.57	2.57	2.57	2.57	2.58	2.58	2.58	2.58	2.58	2.58	2.58	2.58	2.58
2500	3.24	3.24	3.25	3.25	3.25	3.26	3.26	3.27	3.27	3.27	3.28	3.28	3.28	3.29	3.29	3.29	3.29	3.29	3.29
3000	3.95	3.95	3.96	3.97	3.97	3.98	3.98	3.99	4.00	4.00	4.01	4.01	4.01	4.02	4.02	4.02	4.02	4.02	4.02
3500	4.68	4.69	4.70	4.71	4.71	4.72	4.73	4.74	4.75	4.75	4.76	4.76	4.77	4.77	4.78	4.78	4.78	4.78	4.78
4000	5.44	5.45	5.46	5.47	5.48	5.49	5.50	5.51	5.52	5.53	5.54	5.55	5.55	5.56	5.57	5.57	5.57	5.57	5.57
4500	6.22	6.24	6.25	6.26	6.26	6.28	6.29	6.30	6.31	6.33	6.34	6.35	6.36	6.37	6.38	6.38	6.39	6.39	6.39
5000	7.04	7.05	7.07	7.08	7.10	7.11	7.13	7.14	7.16	7.17	7.19	7.20	7.21	7.22	7.23	7.23	7.24	7.24	7.24
5500	7.88	7.89	7.91	7.93	7.95	7.97	7.99	8.01	8.02	8.04	8.06	8.07	8.08	8.09	8.10	8.11	8.12	8.12	8.12
6000	8.75	8.77	8.79	8.81	8.83	8.85	8.87	8.90	8.92	8.94	8.96	8.98	8.99	9.01	9.02	9.03	9.03	9.04	9.04
6500	9.65	9.67	9.70	9.72	9.74	9.77	9.80	9.82	9.85	9.87	9.89	9.91	9.93	9.95	9.96	9.98	9.98	9.99	9.99
7000	10.59	10.61	10.64	10.66	10.69	10.72	10.75	10.78	10.81	10.84	10.87	10.91	10.93	10.95	10.96	10.97	10.98	10.98	10.98
7500	11.56	11.59	11.61	11.64	11.67	11.71	11.74	11.78	11.81	11.84	11.88	11.91	11.93	11.96	11.98	11.99	12.01	12.01	12.02
8000	12.57	12.60	12.63	12.66	12.69	12.73	12.77	12.81	12.85	12.89	12.93	12.96	13.00	13.02	13.05	13.07	13.08	13.09	13.09
8500	13.62	13.65	13.68	13.71	13.75	13.79	13.84	13.88	13.93	13.98	14.02	14.06	14.10	14.14	14.17	14.19	14.21	14.22	14.22
9000	14.71	14.74	14.77	14.81	14.85	14.90	14.95	15.00	15.06	15.11	15.16	15.21	15.26	15.30	15.34	15.36	15.38	15.40	15.40
9500	15.84	15.87	15.90	15.95	15.99	16.05	16.11	16.17	16.23	16.29	16.36	16.42	16.47	16.52	16.56	16.59	16.62	16.63	16.64
10000	17.02	17.05	17.08	17.13	17.18	17.24	17.31	17.38	17.46	17.53	17.60	17.67	17.74	17.80	17.85	17.89	17.92	17.94	17.94
10500	18.25	18.27	18.31	18.36	18.42	18.49	18.57	18.65	18.74	18.83	18.91	19.00	19.07	19.14	19.20	19.25	19.29	19.31	19.32
11000	19.53	19.55	19.59	19.64	19.71	19.79	19.88	19.98	20.08	20.18	20.29	20.39	20.48	20.57	20.64	20.70	20.74	20.77	20.78
11500	20.86	20.88	20.92	20.98	21.06	21.15	21.25	21.37	21.49	21.61	21.74	21.86	21.97	22.07	22.16	22.23	22.29	22.32	22.33
12000	22.23	22.26	22.30	22.37	22.46	22.57	22.69	22.83	22.97	23.12	23.27	23.41	23.55	23.67	23.78	23.87	23.93	23.97	23.99
12500	23.66	23.69	23.74	23.82	23.92	24.05	24.20	24.36	24.53	24.71	24.89	25.06	25.22	25.37	25.50	25.61	25.69	25.74	25.75
13000	25.13	25.16	25.23	25.33	25.45	25.60	25.77	25.97	26.17	26.38	26.60	26.81	27.01	27.19	27.35	27.48	27.58	27.64	27.67
13500	26.62	26.68	26.77	26.88	27.03	27.21	27.42	27.65	27.89	28.15	28.41	28.67	28.92	29.15	29.35	29.52	29.65	29.73	29.75
14000	28.15	28.23	28.34	28.49	28.67	28.89	29.14	29.41	29.71	30.03	30.35	30.67	30.99	31.29	31.55	31.77	31.94	32.05	32.08
14500	29.71	29.82	29.96	30.15	30.37	30.63	30.93	31.27	31.64	32.03	32.44	32.86	33.27	33.67	34.02	34.32	34.56	34.70	34.75
15000	31.30	31.44	31.63	31.85	32.13	32.45	32.82	33.24	33.70	34.20	34.73	35.29	35.85	36.39	36.89	37.33	37.68	37.90	37.97
15500	32.92	33.11	33.33	33.62	33.95	34.35	34.81	35.34	35.93	36.59	37.30	38.06	38.86	39.68	40.47	41.20	41.80	42.21	42.35
16000	34.58	34.81	35.09	35.44	35.86	36.35	36.93	37.60	38.38	39.27	40.27	41.41	42.70	44.17	45.91	48.26			
16500	36.27	36.56	36.91	37.33	37.85	38.47	39.21	40.10	41.15	42.42	43.98	46.01	49.31						
17000	38.01	38.36	38.79	39.31	39.95	40.74	41.70	42.90	44.43	46.47	49.76								
17500	39.78	40.21	40.73	41.38	42.18	43.20	44.50	46.22	48.73										
18000	41.61	42.12	42.76	43.56	44.58	45.93	47.77												
18500	43.48	44.11	44.88	45.88	47.21	49.07													
19000	45.42	46.17	47.13	48.39															
19500	47.44	48.34	49.52																
20000	49.53	48.67																	

APPENDIX E

SAMPLE SIMULATION

A sample problem is included to display a representative input data set and the output that can be expected from the MCSP. A description of the desired simulation follows:

Number of Monte-Carlo runs - 100,

Track has 233 points with 4-Hz, sampling rate with
approximate velocity data available,

Direction of approach - 45°,

Order of filter - 6,

Initialization by synthetic measurements is desired,

Adaptive \underline{Q} with $K_1 = .9$, $K_2 = .1$,

Noise statistics are $\sigma_R = 5.0 + .001 \times \text{Range}$

$\sigma_B = \sigma_E = .002$ radians,

Shell flight times are available, and shell-at-target
accuracy calculations are desired,

$\underline{\phi}$, $\underline{\Gamma}$, and \underline{H} are standard 6th order matrices,

Desired output includes all available plots.

An abridged listing of the input data appears next
followed by representative portions of the output produced.


```

10 0.54388831170 04 0.0 0.43026909950 04
10 -0.130+9261820 03 0.0 -0.27150069810 02
** ** ** **

```

TRACK DATA CONTINUES IN SAME FCRMAT

```

0.51702916290 04 -0.14898673760 04 0.183223115360 04
-0.22767851600 03 -0.930786678660 02 0.43371113810 02
-0.52283387310 04 -0.15128732770 04 0.18430978380 04
-0.22962578050 03 -0.88165457970 02 0.43371113810 02
-0.52868667220 04 -0.15346267080 04 0.185338716490 04
-0.23146703370 03 -0.83211133920 02 0.43371113810 02
-0.53+58485810 04 -0.155511176290 04 0.18646317260 02
-0.23320142540 03 -0.73219193550 02 0.43371113810 02
11920540 02
0.11370800 02
0.110813100 02
0.110247610 02
0.109574520 02
0.109093590 02
0.108506220 02
0.107914330 02
0.107357050 02
0.106753570 02
** ** ** **

```

SHELL FLIGHT TIMES CONTINUE IN SAME FCRMAT

```

0.816023150 01
0.820923710 01
0.8337852630 01
0.846804230 01
0.859773040 01
** ** ** **

```

END OF DATA SET FOR THIS EXAMPLE

ADAPTIVE - THESIS SAMPLE
ORDER OF FILTER= 0

NUMBER OF MONTE CARLO PUNS= 100

CCOE FOR DIRECTION OF APPROACH= 1

TRANSLATION OF X DATA PCINIS= 0.0
Y DATA PCINIS= 0.0
Z DATA PCINIS= 0.0

NCISE CODE= 1

PHI MATRIX

(1,1)=	1.0000000000000000	00	(1,2)=	2.5000000000000000	-01
(1,3)=	0.0		(1,4)=	0.0	
(1,5)=	0.0		(1,6)=	1.0000000000000000	00
(2,1)=	0.0		(2,2)=	0.0	
(2,3)=	0.0		(2,4)=	0.0	
(2,5)=	0.0		(2,6)=	0.0	
(3,1)=	1.0000000000000000	00	(3,2)=	0.0	
(3,3)=	0.0		(3,4)=	2.5000000000000000	-01
(3,5)=	0.0		(3,6)=	0.0	
(4,1)=	0.0		(4,2)=	0.0	
(4,3)=	0.0		(4,4)=	1.0000000000000000	00
(4,5)=	0.0		(4,6)=	0.0	
(5,1)=	0.0		(5,2)=	0.0	
(5,3)=	1.0000000000000000	00	(5,4)=	2.5000000000000000	-01
(5,5)=	0.0		(5,6)=	0.0	
(6,1)=	0.0		(6,2)=	0.0	
(6,3)=	0.0		(6,4)=	1.0000000000000000	00
(6,5)=	0.0		(6,6)=	0.0	

H CR MEASUREMENT MATRIX

(1,1)=	1.0000000000000000	00	(1,2)=	0.0	
(1,3)=	0.0		(1,4)=	0.0	
(1,5)=	0.0		(1,6)=	0.0	
(2,1)=	0.0		(2,2)=	0.0	
(2,3)=	1.0000000000000000	00	(2,4)=	0.0	
(2,5)=	0.0		(2,6)=	0.0	
(3,1)=	0.0		(3,2)=	0.0	
(3,3)=	1.0000000000000000	00	(3,4)=	0.0	
			(3,6)=	0.0	

GAMMA MATRIX

(1,1)=	0.0		(1,2)=	0.0	
(1,3)=	0.0		(2,2)=	0.0	
(2,1)=	0.0		(3,2)=	0.0	
(2,3)=	0.0		(4,2)=	0.0	
(3,1)=	0.0		(5,2)=	0.0	
(3,3)=	0.0		(6,2)=	0.0	
(4,1)=	0.0				
(4,3)=	0.0				
(5,1)=	0.0				
(5,3)=	0.0				
(6,1)=	0.0				
(6,3)=	0.0				

P(K/K-1) OR PREOICION COVARIANCE MATRIX

```

(1,1)= 0.0
(1,2)= 0.0
(1,3)= 0.0
(1,4)= 0.0
(1,5)= 0.0
(1,6)= 0.0
(2,1)= 0.0
(2,2)= 0.0
(2,3)= 0.0
(2,4)= 0.0
(2,5)= 0.0
(2,6)= 0.0
(3,1)= 0.0
(3,2)= 0.0
(3,3)= 0.0
(3,4)= 0.0
(3,5)= 0.0
(3,6)= 0.0
(4,1)= 0.0
(4,2)= 0.0
(4,3)= 0.0
(4,4)= 0.0
(4,5)= 0.0
(4,6)= 0.0
(5,1)= 0.0
(5,2)= 0.0
(5,3)= 0.0
(5,4)= 0.0
(5,5)= 0.0
(5,6)= 0.0
(6,1)= 0.0
(6,2)= 0.0
(6,3)= 0.0
(6,4)= 0.0
(6,5)= 0.0
(6,6)= 0.0

```

INITIAL COVARIANCES OF RANDOM FORCING MATRIX OF
STATE EXCIATION FOR ADAPTIVE Q FILTER

WEIGHTING FACTORS FOR C MATRIX UPDATE ARE - K1= 0.9000000 00
K2= 0.1000000 00

```

(1,1)= 0.0
(1,2)= 0.0
(1,3)= 0.0
(2,1)= 0.0
(2,2)= 0.0
(2,3)= 0.0
(3,1)= 0.0
(3,2)= 0.0
(3,3)= 0.0

```

R CR COVARIANCE OF MEAS. ERROR MATRIX GENERATED BY PROGRAM

RANGE STAN DEV,RR= 5.0000000000000 00
BEARING STAN DEV,RB= 2.0000000000000-03
ELEVATION STAN DEV,RE= 2.0000000000000-03

THE FIRST 2 ESTIMATES OF POSITION WILL BE THE OBSERVED POSITION

STATE INITIAL CONOITIONS ARE

```

(1,1)= 4.0727172264900 03
(2,1)= 0.0
(3,1)= 4.0632459995820 03
(4,1)= 0.0
(5,1)= 4.3510848373670 03
(6,1)= 0.0

```


POINT NUMBER	TRUE TRACK POINTS	FILTER PRED POINTS	RESIDUALS	TIME OF FLT TRACK TIME WHEN FIRED	MEAN MISS DISTANCE
37	X = 0.3264150 04	0.3262890 04	-0.1257670 01	0.896300 01	
	Y = 0.3264150 04	0.2903960 04	-0.1250170 04	---	0.1515640 04
	Z = 0.3869000 04	0.4711080 04	0.8420770 03	0.114820 01	
38	X = 0.3244460 04	0.3204730 04	-0.3972560 02	0.889610 01	
	Y = 0.3244460 04	0.3170250 04	-0.17420350 02	---	0.4169030 03
	Z = 0.3845640 04	0.4255960 04	0.4063180 03	0.2241540 01	
39	X = 0.3224580 04	0.3056660 04	-0.1681180 03	0.883010 01	
	Y = 0.3224580 04	0.2826990 04	-0.3678910 03	---	0.6507390 03
	Z = 0.3822060 04	0.4308930 04	0.4868760 03	0.367960 01	
40	X = 0.3204520 04	0.3059470 04	-0.1450480 03	0.876340 01	
	Y = 0.3204520 04	0.2331720 04	-0.8727990 03	---	0.1149160 04
	Z = 0.3798250 04	0.4551560 04	0.1533150 03	0.494630 01	
41	X = 0.3184270 04	0.2952780 04	-0.2314850 03	0.869610 01	
	Y = 0.3184270 04	0.2129550 04	-0.1034980 04	---	0.1343100 04
	Z = 0.3774220 04	0.4598700 04	0.8244780 03	0.621540 01	
42	X = 0.3163630 04	0.2937710 04	-0.3261210 03	0.862820 01	
	Y = 0.3163630 04	0.2323370 04	-0.1620450 04	---	0.1355600 04
	Z = 0.3749970 04	0.45553400 04	0.8054320 03	0.748700 01	
43	X = 0.3143210 04	0.2787870 04	-0.3553350 03	0.855970 01	
	Y = 0.3143210 04	0.2119970 04	-0.3232420 03	---	0.1241530 04
	Z = 0.3725450 04	0.4475670 04	0.1501790 03	0.876120 01	
44	X = 0.3122400 04	0.2708350 04	-0.4140490 03	0.849060 01	
	Y = 0.3122400 04	0.2244650 04	-0.8777060 03	---	0.1170760 04
	Z = 0.3700800 04	0.4555680 04	0.8548860 03	0.100380 02	

***** PERFORMANCE TABLE OUTOUT CONTINUES FOR REMAINING TRACK POINTS *****

AVERAGE SHELL AT TARGET MISS DISTANCE = 0.182777740 03 YARDS

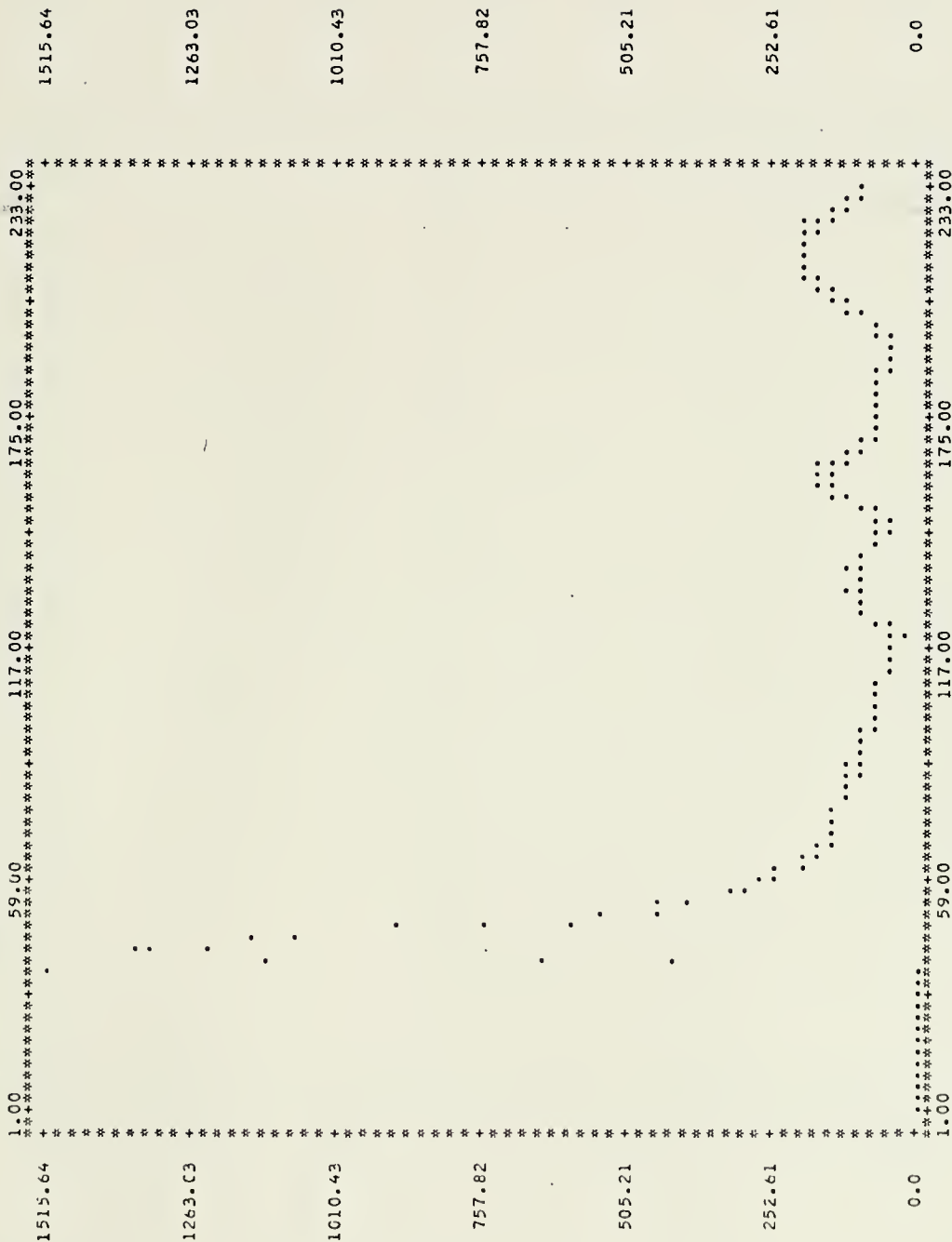
PERFORMANCE FACTOR OF FILTER, AVERAGE FILTER POSITION ESTIMATE ERROR = 0.265126090 01 YARDS

AVERAGE FILTER VELOCITY ESTIMATE ERROR = 0.228808250 02 YARDS/SEC

POINT NO	POSITION ESTIMATE ERRORS	POSITION VARIANCES	VELOCITY /ERRORS	VELOCITY VARIANCES AROUND ESTIMATE/ AROUND TRUTH
1	X: -0.5869300 00 Y: 0.8871200 00 Z: 0.2667850 01	X: 0.1264470 03 Y: 0.1589200 03 Z: 0.1409170 03	0.9251410 01 -0.1307940 03 0.4167560 02	X: 0.1000710 03 Y: 0.5753950 03 Z: 0.4003770 02
2	X: 0.7545570 00 Y: 0.1257540 01 Z: 0.8300160 00	X: 0.1515110 03 Y: 0.1315970 03 Z: 0.1596330 03	0.5350410 01 0.14933670 01 -0.5833430 01	X: 0.4147960 04 Y: 0.4641090 04 Z: 0.4518960 04
3	X: 0.7057070 00 Y: 0.1470990 01 Z: -0.3008100 01	X: 0.5628680 03 Y: 0.5842220 03 Z: 0.5516330 03	-0.8456680 01 -0.3404860 02 0.3861960 01	X: 0.3755090 04 Y: 0.8318880 04 Z: 0.5082250 04
4	X: 0.1747370 01 Y: -0.9144870 01 Z: 0.1500260 00	X: 0.5450270 03 Y: 0.2939770 03 Z: 0.7351480 03	-0.5625930 01 -0.8527660 02 0.3149020 02	X: 0.1923290 04 Y: 0.9721730 04 Z: 0.2291400 04
5	X: 0.6144890 01 Y: -0.1184530 02 Z: 0.6170080 01	X: 0.3272940 03 Y: 0.1837750 03 Z: 0.5795450 03	-0.8143510 01 -0.1035920 03 0.4550990 02	X: 0.1897520 04 Y: 0.1131200 05 Z: 0.2083000 04
6	X: 0.7955340 01 Y: -0.1154870 02 Z: 0.9196870 01	X: 0.1883330 03 Y: 0.2499670 03 Z: 0.5278870 03	-0.1626740 02 -0.1058320 03 0.4551000 02	X: 0.2271090 04 Y: 0.1567610 05 Z: 0.22319410 04
7	X: 0.7656690 01 Y: -0.8484210 01 Z: 0.9714840 01	X: 0.1438700 03 Y: 0.1250970 03 Z: 0.3943640 03	-0.2684820 02 -0.1066440 03 0.4398590 02	X: 0.1768330 04 Y: 0.1290470 05 Z: 0.1874850 04
8	X: 0.6205060 01 Y: -0.4205540 01 Z: 0.8757070 01	X: 0.1343870 03 Y: 0.1444630 03 Z: 0.5234990 03	-0.2990000 02 -0.9361390 02 0.3855810 02	X: 0.2034090 04 Y: 0.1340990 05 Z: 0.1934440 04
9	X: 0.3108800 01 Y: -0.1244540 01 Z: 0.7188760 01	X: 0.1365750 03 Y: 0.1237080 03 Z: 0.1886110 03	-0.3800480 02 -0.9702130 02 0.3895310 02	X: 0.2036340 04 Y: 0.1147520 05 Z: 0.1995420 04
10	X: 0.1329180 01 Y: -0.1445250 01 Z: 0.4942810 01	X: 0.9378140 02 Y: 0.1117670 03 Z: 0.1326700 03	-0.3746770 02 -0.9168930 02 0.3280510 02	X: 0.1835920 04 Y: 0.9681160 04 Z: 0.1624800 04

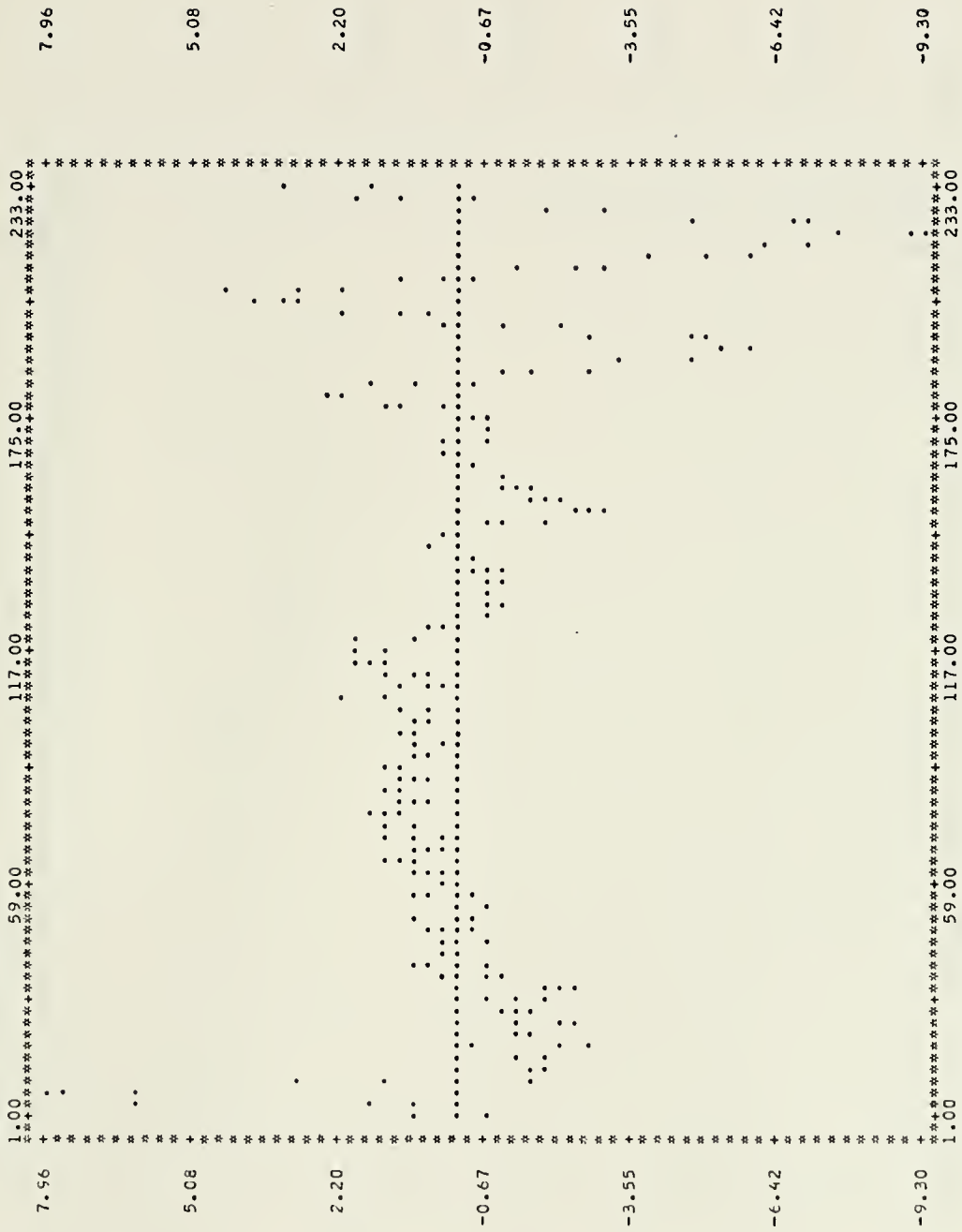
***** FILTER ERROR TABLE CONTINUES FOR REMAINING TRACK POINTS *****

KETRON MK-86 THESIS MEAN MISS DISTANCE WHEN SHELL ARRIVES IN YDS.



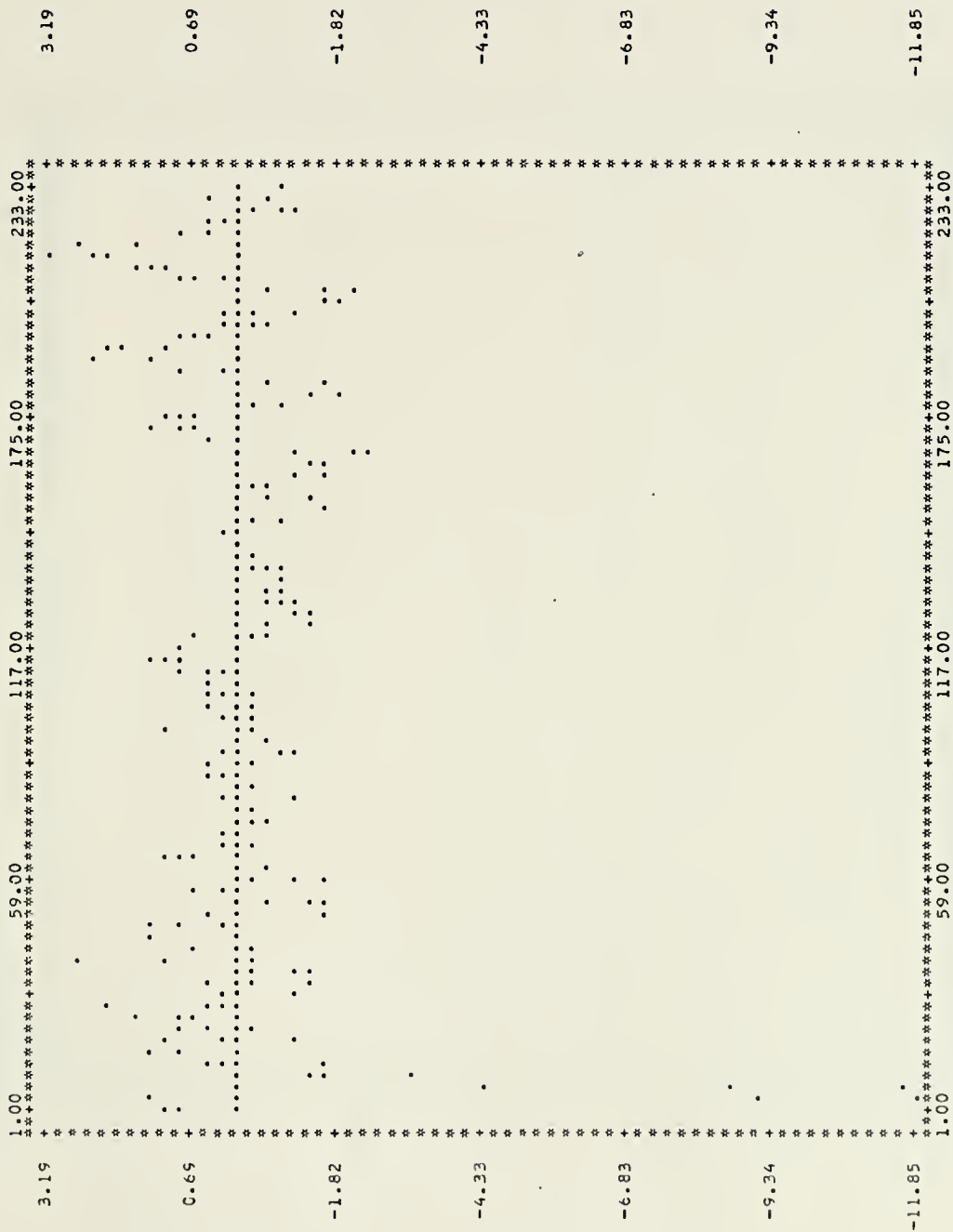
X-SCALE: "X"= 0.290E 01 UNITS
 Y-SCALE: "Y"= 0.251E 02 UNITS

KEIRON MK-86 THESIS
MEAN ERROR IN X POSITION



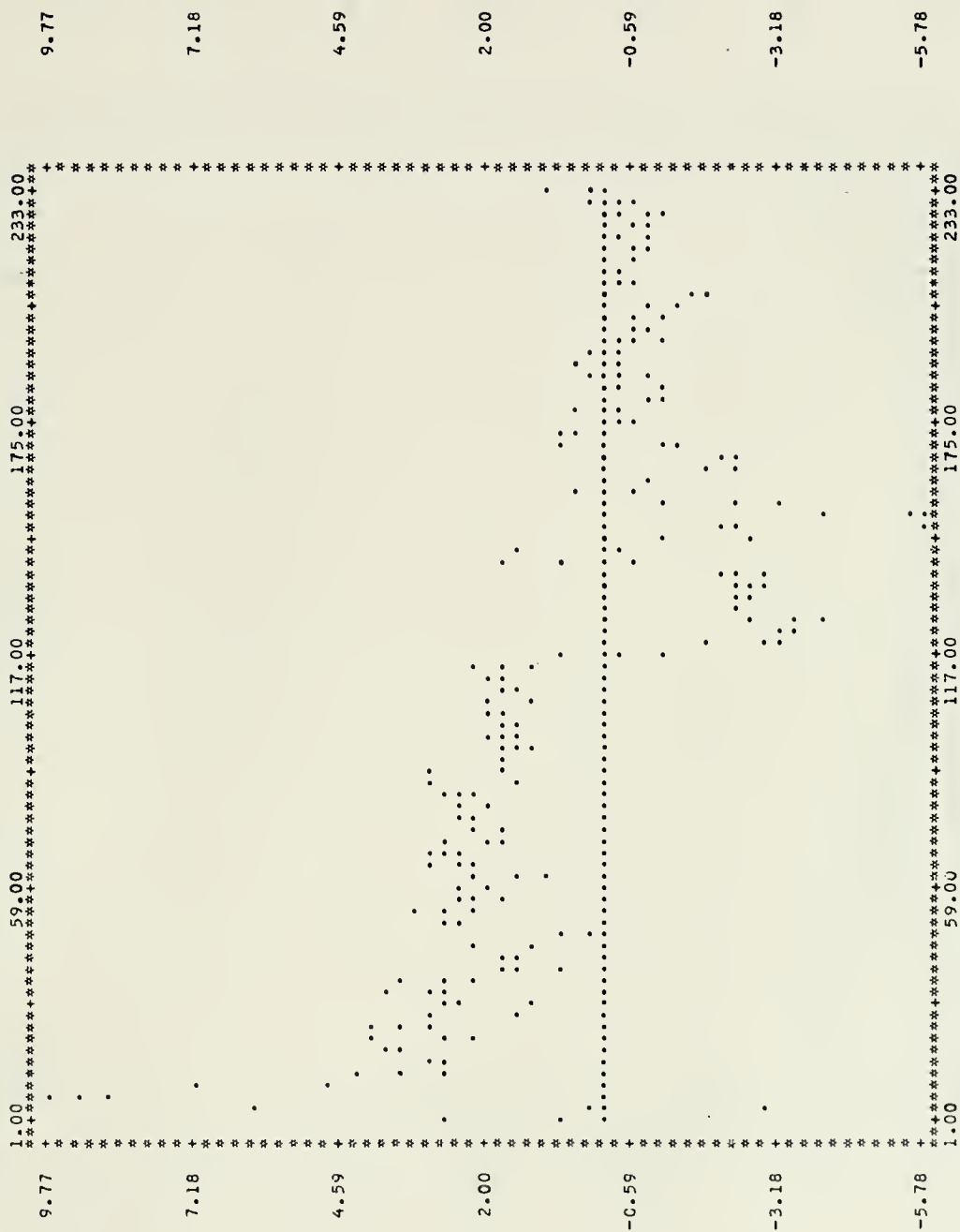
X-SCALE: "X"= 0.290E 01 UNITS
Y-SCALE: "Y"= 0.288E 00 UNITS

KETRON MK-86 THESIS
MEAN ERROR IN Y POSITION



X-SCALE: **N= 0.290E 01 UNITS
Y-SCALE: **N= 0.251E 00 UNITS

KETRON MK-86 THESIS
MEAN ERROR IN Z POSITION



X-SCALE: "X"= 0.290E 01 UNITS
Y-SCALE: "Y"= 0.259E 00 UNITS

KETRON MK-86 THESIS
X POSITION VARIANCE



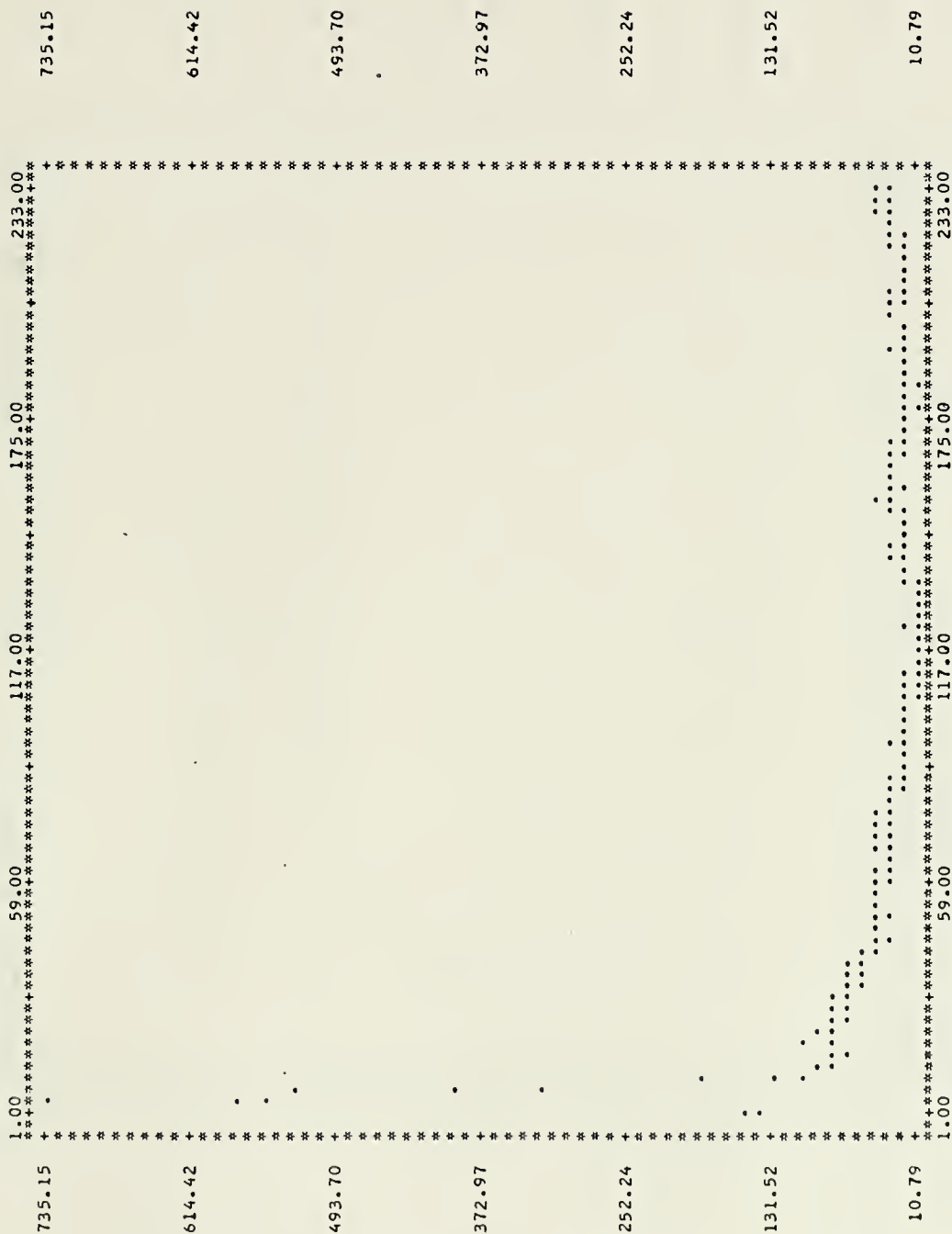
X-SCALE: "*"= 0.250E 01 UNITS
Y-SCALE: "*"= 0.937E 01 UNITS

KETRON MK-86 THESIS
Y POSITION VARIANCE



X-SCALE: ""= 0.290E 01 UNITS
Y-SCALE: ""= 0.973E 01 UNITS

KETRON MK-86 THESIS
Z POSITION VARIANCE



X-SCALE: "X"= 0.290E 01 UNITS

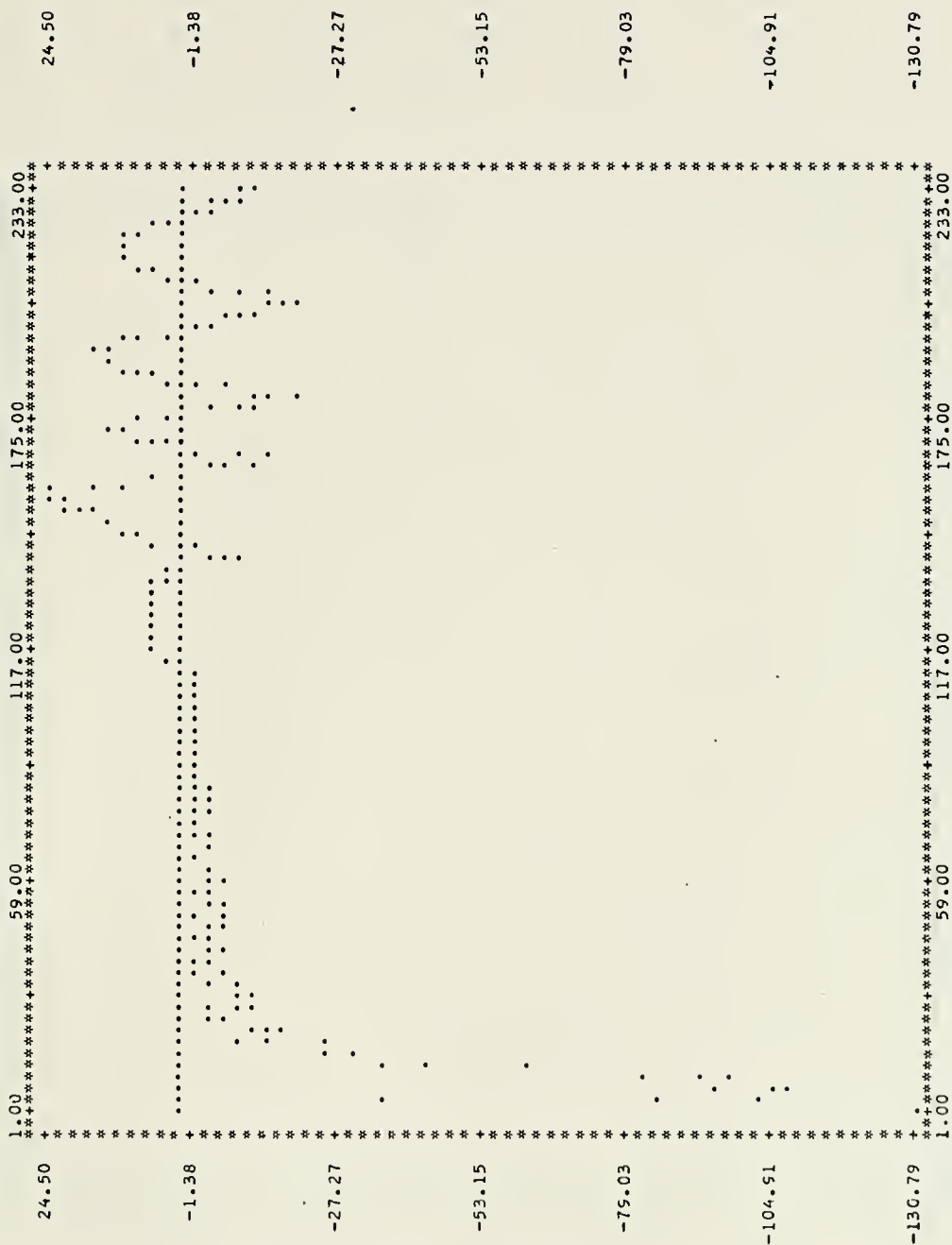
Y-SCALE: "Y"= 0.121E 02 UNITS

KETRON MK-86 THESIS
MEAN VELOCITY ERROR IN X DIRECTION



X-SCALE: "*"= 0.290E 01 UNITS
Y-SCALE: "*"= 0.130E 01 UNITS

KEIRON MK-86 THESIS
MEAN VELOCITY ERROR IN Y DIRECTION



X-SCALE: "*"= 0.290E 01 UNITS
Y-SCALE: "*"= 0.259E 01 UNITS

KETRON MK-86 THESIS
MEAN VELOCITY ERROR IN Z DIRECTION



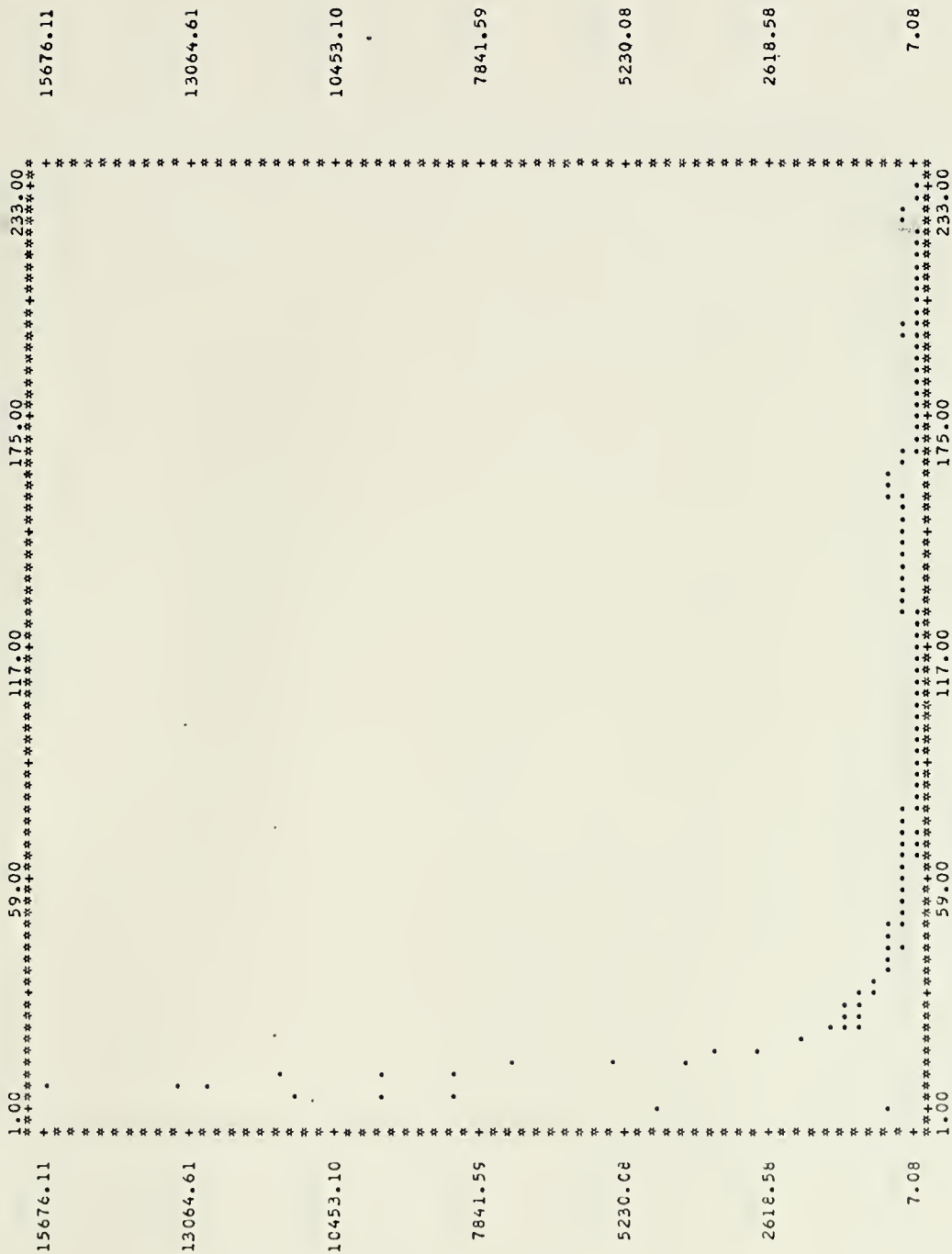
X-SCALE: "X"= 0.290E 01 UNITS
Y-SCALE: "Y"= 0.182E 01 UNITS

KETRON WK-86 THESIS X VELOCITY VARIANCE



X-SCALE: "X"= 0.290E 01 UNITS
Y-SCALE: "Y"= 0.691E 02 UNITS

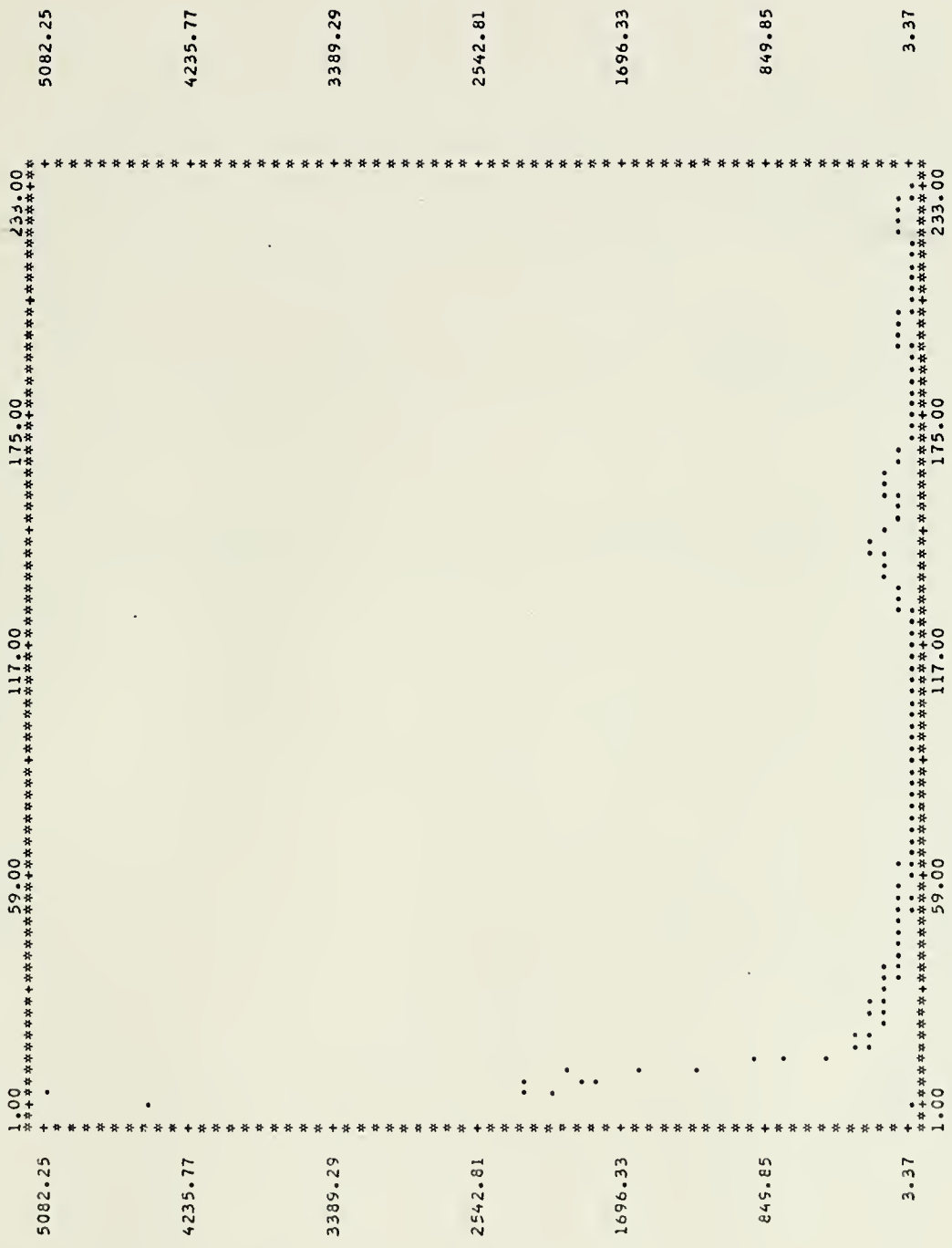
KETRON MK-86 THESIS Y VELOCITY VARIANCE



X-SCALE: "*" = 0.290E 01 UNITS

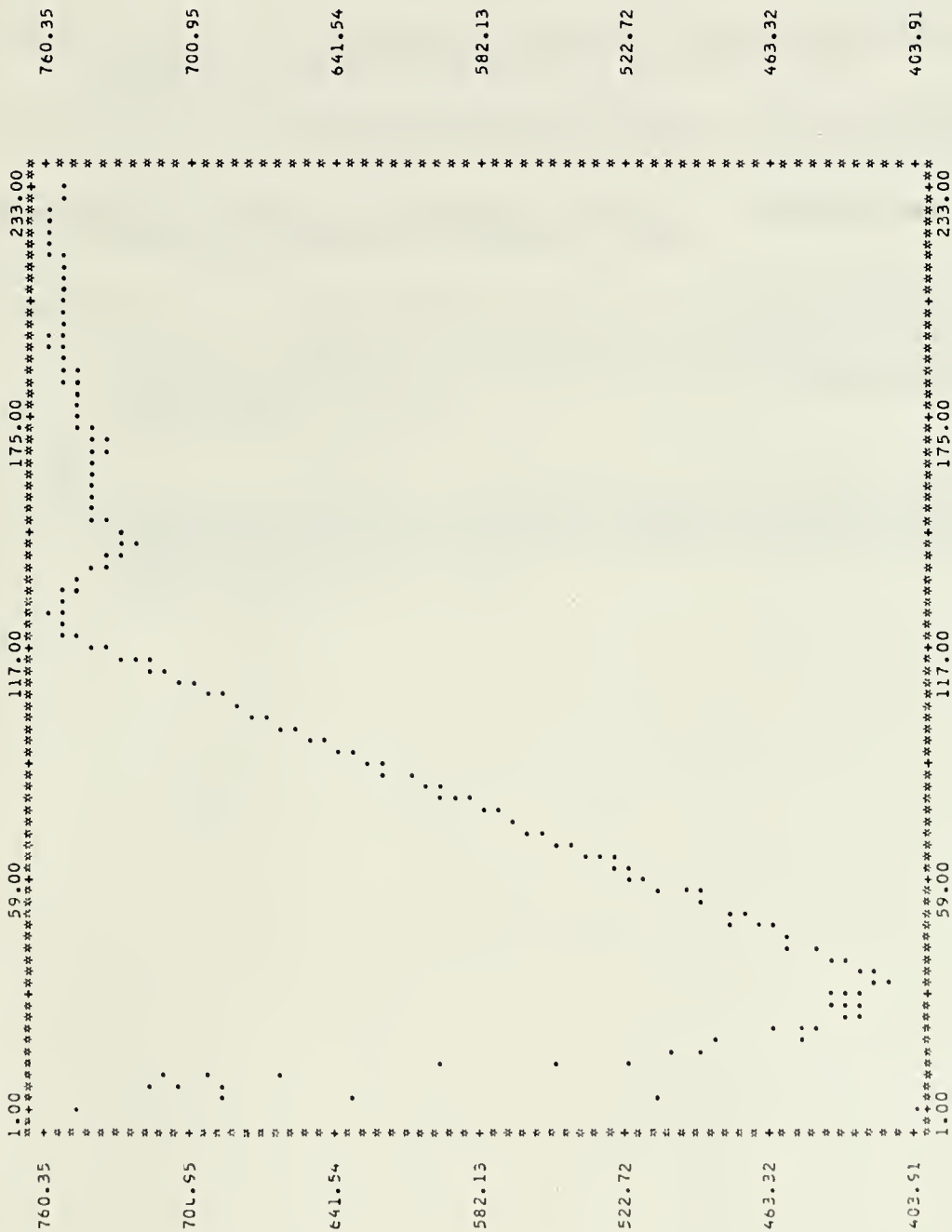
Y-SCALE: "*" = 0.261E 03 UNITS

KETRON MK-86 THESIS
Z VELOCITY VARIANCE



X-SCALE: ""= 0.290E 01 UNITS
Y-SCALE: ""= 0.646E 02 UNITS

KRTFON YK-86 THESIS TARGET SPEED IN FT/SEC



X-SCALE: "X"= 0.290E 01 UNITS
Y-SCALE: "Y"= 0.554E 01 UNITS

BIBLIOGRAPHY

1. Meditch, J. S., Stochastic Optimal Linear Estimation and Control, McGraw-Hill Book Co., Inc., 1969.
2. Korn, G. A., Random-Process Simulation and Measurements, McGraw-Hill Book Co., Inc., 1966.
3. Engineering Design Handbook, Section 1, Headquarters, U.S. Army Materiel Command, 1969.
4. Aldrich, G. T. and Krabill, W. B., "An Application of Kalman Techniques to Aircraft and Missile Radar Tracking," American Institute of Aeronautics and Astronautics Journal, v. 11, No. 7, p. 932-938, July 1973.
5. Will, T. J., A FORTRAN Simulation of the Target Data Filtering Section of the AA Tracking Loop in the MK-86 GFCS, Master's Thesis, Naval Postgraduate School, Monterey, 1973.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2. Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3. Department Chairman, Code 52 Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	2
4. AssocProfessor D. E. Kirk, Code 52 Ki Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	2
5. LTJG Michael Gordon Ketron NCS Box 33-B363 FPO New York, N. Y. 09540	1
6. LT D. F. Regener, USN Naval Ship Missile System Engineering Station Space City 8 Port Hueneme, California 93043	2



94831 S
7 JUN 76
7 JUN 76

23472
23646

Thesis

152415

K3953 Ketron

c.1

Monte-Carlo evaluation
of digital filters for
fire control systems.

94831 S
7 JUN 76
7 JUN 76

23472
23646

Thesis

152415

K3953 Ketron

c.1

Monte-Carlo evaluation
of digital filters for
fire control systems.

thesK3953

Monte-Carlo evaluation of digital filter



3 2768 001 03222 0

DUDLEY KNOX LIBRARY